Documentação do Trident 25.10

Trident

NetApp
February 25, 2026

This PDF was generated from https://docs.netapp.com/pt-br/trident-2510/index.html on February 25, 2026. Always check docs.netapp.com for the latest.

Índice

Documentação do Trident 25.10 . . . 1
Notas de lançamento . . . 2
O que há de novo . . . 2
O que há de novo em 25.10 . . . 2
Alterações em 25.06.2 . . . 4
Alterações em 25.06.1 . . . 4
Alterações em 25.06 . . . 4
Alterações em 25.02.1 . . . 7
Alterações em 25.02 . . . 7
Alterações em 24.10.1 . . . 9
Alterações em 24.10 . . . 9
Alterações em 24.06 . . . 11
Alterações em 24.02 . . . 12
Alterações em 23.10 . . . 12
Alterações em 23.07.1 . . . 13
Alterações em 23.07 . . . 13
Alterações em 23.04 . . . 14
Alterações em 23.01.1 . . . 15
Alterações em 23.01 . . . 15
Alterações em 22.10 . . . 16
Alterações em 22.07 . . . 17
Alterações em 22.04 . . . 18
Alterações em 22.01.1 . . . 19
Alterações em 22.01.0 . . . 19
Alterações em 21.10.1 . . . 20
Alterações em 21.10.0 . . . 20
Problemas conhecidos . . . 21
Encontre mais informações . . . 22
Versões anteriores da documentação . . . 22
NetApp Trident suporte para sistemas de storage ONTAP ASA r2 . . . 23
Operações suportadas . . . 23
Operações não suportadas . . . 23
Problemas conhecidos . . . 23
A restauração de backups Restic de arquivos grandes pode falhar . . . 24
Comece agora . . . 25
Saiba mais sobre Trident . . . 25
Saiba mais sobre Trident . . . 25
Arquitetura do Trident . . . 26
Conceitos . . . 29
Início rápido para Trident . . . 33
Qual é o próximo passo? . . . 34
Requisitos . . . 34
Informações essenciais sobre Trident . . . 34

Frontends suportados (orquestradores) . . . 34
Backends suportados (storage) . . . 35
Suporte do Trident para KubeVirt e OpenShift Virtualization . . . 35
Requisitos de funcionalidade . . . 36
Sistemas operacionais de host testados . . . 36
Configuração do host . . . 37
Configuração do sistema de storage . . . 37
Portas Trident . . . 37
Imagens de contêiner e versões correspondentes do Kubernetes . . . 37
Instale Trident . . . 39
Instale usando o operador Trident . . . 39
Instalar usando o tridentctl . . . 39
Instale usando o operador certificado OpenShift . . . 39
Usar o Trident . . . 40
Prepare o nó de trabalho . . . 40
Selecionando as ferramentas certas . . . 40
Descoberta de serviço de nó . . . 40
Volumes NFS . . . 41
volumes iSCSI . . . 41
Volumes NVMe/TCP . . . 45
Volumes SCSI over FC . . . 46
Prepare-se para provisionar volumes SMB . . . 49
Configurar e gerenciar backends . . . 50
Configurar backends . . . 50
Azure NetApp Files . . . 51
Google Cloud NetApp Volumes . . . 69
Configurar um NetApp HCI ou SolidFire backend . . . 86
Drivers SAN do ONTAP . . . 91
Drivers NAS do ONTAP . . . 121
Amazon FSx for NetApp ONTAP . . . 158
Criar backends com kubectl . . . 193
Gerenciar backends . . . 200
Criar e gerenciar classes de armazenamento . . . 210
Crie uma storage class . . . 210
Gerenciar classes de armazenamento . . . 213
Provisionar e gerenciar volumes . . . 215
Provisionar um volume . . . 215
Expandir volumes . . . 219
Importar volumes . . . 230
Personalize os nomes e rótulos dos volumes . . . 240
Compartilhe um volume NFS entre namespaces . . . 243
Clonar volumes entre namespaces . . . 247
Replicar volumes usando SnapMirror . . . 250
Usar a topologia CSI . . . 256
Trabalhar com Snapshots . . . 264

Trabalhar com Snapshots de grupo de volume . . . 272
Gerencie e monitore Trident . . . 277
Atualizar Trident . . . 277
Atualizar Trident . . . 277
Atualize com o operador . . . 278
Atualizar com tridentctl . . . 283
Gerencie Trident usando tridentctl . . . 283
Comandos e flags globais . . . 283
Opções e parâmetros de comando . . . 285
Suporte a plugin . . . 291
Monitorar Trident . . . 291
Visão geral . . . 291
Etapa 1: defina um alvo do Prometheus . . . 291
Passo 2: criar um Prometheus ServiceMonitor . . . 292
Etapa 3: Consultar métricas do Trident com PromQL . . . 294
Saiba mais sobre a telemetria Trident AutoSupport . . . 295
Desativar métricas do Trident . . . 296
Desinstalar Trident . . . 296
Determine o método de instalação original . . . 296
Desinstalar uma instalação do Trident . . . 296
Desinstalar uma `tridentctl` instalação . . . 297
Trident for Docker . . . 298
Pré-requisitos para implantação . . . 298
Verifique os requisitos . . . 298
Ferramentas NVMe . . . 300
Ferramentas FC . . . 301
Implantar Trident . . . 303
Método de plugin gerenciado pelo Docker (versão 1.13/17.03 e later) . . . 303
Método tradicional (versão 1.12 ou anterior) . . . 305
Iniciar Trident na inicialização do sistema . . . 306
Atualize ou desinstale Trident . . . 307
Atualizar . . . 308
Desinstalar . . . 309
Trabalhar com volumes . . . 309
Criar um volume . . . 309
Remover um volume . . . 310
Clonar um volume . . . 310
Acessar volumes criados externamente . . . 311
Opções de volume específicas do driver . . . 312
Coletar registros . . . 316
Colete registros para solucionar problemas . . . 317
Dicas gerais de resolução de problemas . . . 317
Gerenciar várias instâncias do Trident . . . 318
Etapas para o plugin gerenciado do Docker (versão 1.13/17.03 ou later) . . . 318
Passos para tradicional (versão 1.12 ou anterior) . . . 318

Opções de configuração de storage . . . 319
Opções globais de configuração . . . 319
Configuração do ONTAP . . . 320
Configuração do software Element . . . 328
Problemas e limitações conhecidos . . . 330
A atualização do Trident Docker Volume Plugin para a versão 20.10 ou posterior a partir de versões mais antigas resulta em falha na atualização com o erro "arquivo ou diretório inexistente". . . . 330
Os nomes dos volumes devem ter no mínimo 2 caracteres de comprimento. . . . 331
Docker Swarm possui certos comportamentos que impedem Trident de oferecer suporte a todas as combinações de storage e driver. . . . 331
Se um FlexGroup estiver sendo provisionado, ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup que está sendo provisionado. . . . 331
Melhores práticas e recomendações . . . 332
Implantação . . . 332
Implantar em um namespace dedicado . . . 332
Utilize quotas e limites de intervalo para controlar o consumo de armazenamento . . . 332
Configuração de storage . . . 332
Visão geral da plataforma . . . 332
ONTAP e Cloud Volumes ONTAP melhores práticas . . . 332
Práticas recomendadas do SolidFire . . . 337
Onde encontrar mais informações? . . . 339
Integrar Trident . . . 339
Seleção e implantação de driver . . . 340
Design de storage class . . . 342
Design de pool virtual . . . 343
Operações de volume . . . 344
Serviço de métricas . . . 348
Proteção de dados e recuperação de desastres . . . 349
Replicação e recuperação do Trident . . . 349
Replicação e recuperação de SVM . . . 350
Replicação e recuperação de volume . . . 351
Proteção de dados Snapshot . . . 351
Automatizando o failover de aplicações stateful com Trident . . . 351
Detalhes sobre force detach . . . 351
Detalhes sobre failover automatizado . . . 352
Segurança . . . 357
Segurança . . . 357
Linux Unified Key Setup (LUKS) . . . 358
Criptografia em trânsito Kerberos . . . 365
Proteja aplicativos com Trident Protect . . . 373
Saiba mais sobre Trident Protect . . . 373
Qual é o próximo passo? . . . 373
Instalar Trident Protect . . . 373
Requisitos do Trident Protect . . . 373

Instalar e configurar Trident Protect 377
Instale o plugin Trident Protect CLI 380
Personalizar a instalação do Trident Protect 384
Gerenciar Trident Protect 389
Gerencie a autorização e o controle de acesso do Trident 389
Monitorar recursos do Trident Protect 396
Gerar um pacote de suporte Trident Protect 401
Atualizar Trident Protect 403
Gerenciar e proteger aplicativos 405
Use objetos do Trident Protect AppVault para gerenciar buckets 405
Defina uma aplicação para gestão com Trident Protect 419
Proteja aplicativos usando Trident Protect 423
Restaurar aplicativos 435
Replique aplicações usando NetApp SnapMirror e Trident Protect 453
Migrar aplicativos usando Trident Protect 470
Gerenciar os ganchos de execução do Trident Protect 474
Desinstale Trident Protect 486
Blogs do Trident e Trident Protect 487
Blogs do Trident 487
Blogs do Trident Protect 487
Conhecimento e suporte 489
Perguntas frequentes 489
Perguntas gerais 489
Instale e utilize Trident em um cluster Kubernetes 489
Resolução de problemas e suporte 490
Atualizar Trident 491
Gerenciar backends e volumes 492
Solução de problemas 496
Solução de problemas gerais 496
Implantação malsucedida do Trident usando o operador 497
Implantação malsucedida do Trident usando `tridentctl` 499
Remova completamente Trident e os CRDs 499
Falha no desempacotamento de nó NVMe com namespaces de bloco bruto RWX no Kubernetes 1.26 500
Clientes NFSv4.2 relatam "argumento inválido" após atualizar o ONTAP quando esperam que o "v4.2-xattrs" esteja habilitado 501
Suporte 501
Ciclo de vida do Trident 501
Autoatendimento 502
Suporte da comunidade 502
NetApp suporte técnico da NetApp 502
Para obter mais informações 502
Referência 503
Portas Trident 503
Visão geral 503
API REST Trident 505

Quando usar a API REST . . . 505
Usando API REST . . . 505
Opções de linha de comando . . . 506
Log . . . 506
Kubernetes . . . 506
Docker . . . 507
REST . . . 507
Objetos Kubernetes e Trident . . . 507
Como os objetos interagem entre si? . . . 507
Objetos do Kubernetes `PersistentVolumeClaim` . . . 508
Objetos do Kubernetes `PersistentVolume` . . . 510
Objetos do Kubernetes `StorageClass` . . . 510
Objetos do Kubernetes `VolumeSnapshotClass` . . . 513
Objetos do Kubernetes `VolumeSnapshot` . . . 514
Objetos do Kubernetes `VolumeSnapshotContent` . . . 514
Kubernetes `VolumeGroupSnapshotClass` objetos . . . 515
Objetos do Kubernetes `VolumeGroupSnapshot` . . . 515
Kubernetes `VolumeGroupSnapshotContent` objetos . . . 516
Kubernetes `CustomResourceDefinition` objetos . . . 516
Trident `StorageClass` objetos . . . 516
Objetos de backend Trident . . . 517
Trident `StoragePool` objetos . . . 517
Trident `Volume` objetos . . . 517
Trident `Snapshot` objetos . . . 519
Trident `ResourceQuota` objeto . . . 519
Padrões de Segurança de Pod (PSS) e Restrições de Contexto de Segurança (SCC) . . . 520
Contexto de segurança obrigatório do Kubernetes e campos relacionados . . . 521
Padrões de segurança de pod (PSS) . . . 521
Políticas de segurança de pods (PSP) . . . 522
Restrições de Contexto de Segurança (SCC) . . . 523
Avisos legais . . . 526
Direitos autorais . . . 526
Marcas registradas . . . 526
Patentes . . . 526
Política de privacidade . . . 526
Código aberto . . . 526

Documentação do Trident 25.10

Notas de lançamento

O que há de novo

As notas de versão fornecem informações sobre novos recursos, melhorias e correções de bugs na versão mais recente do NetApp Trident.

> O `tridentctl` binário para Linux fornecido no arquivo zip do instalador é a versão testada e suportada. Esteja ciente de que o `macos` binário fornecido na `/extras` parte do arquivo zip não é testado nem suportado.

O que há de novo em 25.10

Saiba o que há de novo no Trident e Trident Protect, incluindo melhorias, correções e descontinuações.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para Snapshots de grupo de volume CSI com as APIs Kubernetes v1beta1 Volume Group Snapshot para os drivers ONTAP-NAS NFS e ONTAP-SAN-Economy, além do ONTAP-SAN (iSCSI e FC). Veja "Trabalhar com Snapshots de grupo de volume".
  - Adicionado suporte para failover automático de cargas de trabalho com desanexação forçada de volumes para os drivers ONTAP-NAS e ONTAP-NAS-Economy (excluindo SMB em ambos os drivers NAS), e para os drivers ONTAP-SAN e ONTAP-SAN-Economy. Veja "Automatizando o failover de aplicações stateful com Trident".
  - Concorrência de nós Trident aprimorada para maior escalabilidade nas operações de nós para volumes FCP.
  - Adicionado suporte ONTAP AFX para o driver ONTAP NAS. Consulte "Opções e exemplos de configuração do ONTAP NAS".
  - Adicionado suporte para configurar solicitações e limites de recursos de CPU e memória para os contêineres Trident por meio do TridentOrchestrator CR e dos valores do Helm chart. ("Problema #1000", "Problema nº 927", "Problema nº 853", "Problema nº 592", "Problema nº 110").
  - Adicionado suporte FC para a personalidade ASAr2. Veja "Opções e exemplos de configuração do ONTAP SAN".
  - Adicionada opção para servir métricas Prometheus com HTTPS, em vez de HTTP. Veja "Monitorar Trident".
  - Adicionada uma opção `--no-rename` ao importar um volume para manter o nome original, mas permitir que Trident gerencie o ciclo de vida do volume. Veja "Importar volumes".
  - A implantação do Trident agora é executada na classe de prioridade system-cluster-critical.
- Adicionada uma opção para o controlador Trident usar rede via helm, operator e tridentctl ("Problema nº 858").
- Adicionado suporte manual de QoS ao driver ANF, tornando-o pronto para produção no Trident 25.10; esse aprimoramento experimental foi introduzido no Trident 25.06.

Aprimoramentos experimentais

Não deve ser utilizado em ambientes de produção.

- **[Prévia técnica]**: Adicionado suporte para simultaneidade para ONTAP-NAS (somente NFS) e ONTAP-SAN (NVMe para ONTAP 9 unificado), além da prévia técnica existente para o driver ONTAP-SAN (protocolos iSCSI e FCP no ONTAP 9 unificado).

Correções

- **Kubernetes**:
  - Corrigida a inconsistência no nome do contêiner CSI node-driver-registrar, padronizando Linux DaemonSet para node-driver-registrar para corresponder ao Windows DaemonSet e à nomenclatura da imagem do contêiner.
  - Corrigido um problema em que as políticas de exportação para qtrees legados não eram atualizadas corretamente.
- **Openshift**:
  - Corrigido o problema de inicialização do pod do nó Trident em nós Windows no Openshift devido ao SCC ter allowHostDirVolumePlugin definido como falso ("Problema nº 950").
- Corrigido o problema em que o QPS da API do Kubernetes não estava sendo configurado via Helm ("Problema nº 975").
- Corrigida a impossibilidade de montar uma Persistent Volume Claim (PVC) baseada em um Snapshot de uma PVC de sistema de arquivos XFS baseado em NVMe no mesmo nó do Kubernetes.
- Corrigido o problema de alteração do UUID após reiniciar o host/Docker no modo NDVP, adicionando nomes de subsistema exclusivos/compartilhados por backend (por exemplo, netappdvp_subsystem).
- Corrigidos erros de montagem para volumes iSCSI durante a atualização do Trident de versões anteriores à 23.10 para 24.10 e superiores, resolvendo o problema de "SANType inválido".
- Corrigido o problema em que o estado do backend do Trident não estava sendo atualizado para online/offline sem reiniciar o controlador do Trident.
- Corrigida condição de corrida intermitente que causava redimensionamento lento do PVC.
- Corrigido o problema em que os snapshots não eram apagados em clone de volume em caso de falha na clonagem de volumes.
- Corrigida a falha ao remover o volume da fase de preparação quando o caminho do dispositivo era alterado pelo kernel.
- Corrigida a falha ao desestagiar o volume devido ao dispositivo LUKS já estar fechado.
- Corrigido problema em que operações de armazenamento lentas causavam erros ContextDeadline.
- Trident Operator aguardará o k8s-timeout configurável para verificar a versão do Trident.

Trident Protect

NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Adicionadas anotações para controlar os tempos limite de CR de snapshot para CRs de agendamento e

backup:

  - `protect.trident.netapp.io/snapshot-completion-timeout`
  - `protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout`
  - `protect.trident.netapp.io/volume-snapshots-created-timeout`

    Consulte "Anotações de backup e agendamento compatíveis".

- Adicionada uma anotação ao CR de agendamento para configurar o tempo limite de vinculação do PVC, que será usado pelo CR de backup: `protect.trident.netapp.io/pvc-bind-timeout-sec`. Veja "Anotações de backup e agendamento compatíveis".
- Listagens de backup e snapshot aprimoradas com um novo campo para indicar falhas no gancho de execução. `tridentctl-protect`

Alterações em 25.06.2

Trident

Correções

- **Kubernetes**: corrigido um problema crítico em que dispositivos iSCSI incorretos eram detectados ao desanexar volumes de nós do Kubernetes.

Alterações em 25.06.1

Trident

> Para clientes que utilizam SolidFire, por favor, não atualizem para a versão 25.06.1 devido a um problema conhecido ao cancelar a publicação de volumes. A versão 25.06.2 será lançada em breve para corrigir esse problema.

Correções

- **Kubernetes**:
  - Corrigido um problema em que os NQNs não eram verificados antes de serem desvinculados dos subsistemas.
  - Corrigido um problema em que várias tentativas de fechar um dispositivo LUKS levavam a falhas ao desanexar volumes.
  - Corrigida a desativação do volume iSCSI quando o caminho do dispositivo foi alterado desde a sua criação.
  - Clonagem de volumes entre classes de armazenamento.
- **OpenShift**: corrigido um problema em que a preparação do nó iSCSI falhava com o OCP 4.19.
- Aumentou o tempo limite ao clonar um volume usando backends SolidFire ("Problema nº 1008").

Alterações em 25.06

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para snapshots de grupos de volumes CSI com `v1beta1`APIs Kubernetes de Snapshot de grupo de volumes para o driver ONTAP-SAN iSCSI. Consulte "Trabalhar com Snapshots de grupo de volume".

    > VolumeGroupSnapshot é um recurso beta no Kubernetes com APIs beta. Kubernetes 1.32 é a versão mínima necessária para VolumeGroupSnapshot.

  - Adicionado suporte para ONTAP ASA r2 para NVMe/TCP além de iSCSI. Veja "Opções e exemplos de configuração do ONTAP SAN".
  - Adicionado suporte seguro a SMB para volumes ONTAP-NAS e ONTAP-NAS-Economy. Usuários e grupos do Active Directory agora podem ser usados com volumes SMB para maior segurança. Consulte "Ativar SMB seguro".
  - Concorrência de nós Trident aprimorada para maior escalabilidade nas operações de nós para volumes iSCSI.
  - Adicionado `--allow-discards`ao abrir volumes LUKS para permitir comandos discard/TRIM para exigência de espaço.
  - Desempenho aprimorado ao formatar volumes criptografados com LUKS.
  - Limpeza aprimorada do LUKS para dispositivos com falha, mas parcialmente formatados.
  - Idempotência aprimorada do nó Trident para conexão e desconexão de volumes NVMe.
  - Adicionado `internalID` campo à configuração de volume do Trident para o driver ONTAP-SAN-Economy.
  - Adicionado suporte para replicação de volume com SnapMirror para backends NVMe. Consulte "Replicar volumes usando SnapMirror".

Aprimoramentos experimentais

> Não deve ser utilizado em ambientes de produção.

- [Prévia Técnica] Habilitada a execução simultânea de operações do controlador Trident por meio do recurso de sinalização `--enable-concurrency`. Isso permite que as operações do controlador sejam executadas em paralelo, melhorando o desempenho para ambientes ocupados ou de grande porte.

  > Este recurso é experimental e atualmente suporta fluxos de trabalho paralelos limitados com o driver ONTAP-SAN (protocolos iSCSI e FCP).

- [Prévia Técnica] Adicionado suporte manual a QOS com o driver ANF.

Correções

- **Kubernetes**:
  - Corrigido um problema com o CSI NodeExpandVolume em que dispositivos multipath podiam ficar com tamanhos incongruentes quando os discos SCSI subjacentes não estavam disponíveis.
  - Corrigida a falha ao limpar políticas de exportação duplicadas para os drivers ONTAP-NAS e ONTAP-NAS-Economy.

    - Corrigido o problema em que os volumes GCNV usavam NFSv3 por padrão quando `nfsMountOptions` não estava definido; agora, ambos os protocolos NFSv3 e NFSv4 são suportados. Se `nfsMountOptions` não for fornecido, a versão de NFS padrão do host (NFSv3 ou NFSv4) será usada.
    - Problema de implantação corrigido ao instalar Trident usando Kustomize ("Problema nº 831").
    - Corrigidas as políticas de exportação ausentes para PVCs criados a partir de snapshots ("Problema nº 1016").
    - Corrigido o problema em que os tamanhos de volume do ANF não são alinhados automaticamente em incrementos de 1 GiB.
    - Problema resolvido ao usar NFSv3 com Bottlerocket.
- Corrigido problema com volumes ONTAP-NAS-Economy que expandiam até 300 TB apesar de falhas no resize.
- Corrigido problema em que as operações de divisão de clones eram realizadas de forma síncrona ao usar a API REST do ONTAP.

Descontinuações:

- **Kubernetes**: versão mínima do Kubernetes suportada atualizada para v1.27.

Trident Protect

NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Tempos de restauração aprimorados, proporcionando a opção de fazer backups completos com mais frequência.
- Granularidade aprimorada da definição de aplicativos e restauração seletiva com filtragem por Group-Version-Kind (GVK).
- Ressincronização e replicação reversa eficientes ao usar AppMirrorRelationship (AMR) com NetApp SnapMirror, para evitar replicação completa de PVC.
- Adicionada a capacidade de usar a identidade do Pod do EKS para criar AppVault buckets, eliminando a necessidade de especificar um segredo com as credenciais do bucket para clusters EKS.
- Adicionada a capacidade de ignorar a restauração de rótulos e anotações no namespace de restauração, se necessário.
- AppMirrorRelationship (AMR) agora verificará a expansão do PVC de origem e realizará a expansão apropriada no PVC de destino, conforme necessário.

Correções

- Corrigido um bug em que os valores de anotação de snapshots anteriores estavam sendo aplicados a snapshots mais recentes. Todas as anotações de snapshots agora são aplicadas corretamente.
- Definida uma chave secreta para criptografia de movimentação de dados (Kopia / Restic) por padrão, caso não esteja definida.
- Adicionada validação aprimorada e mensagens de erro para a criação do appvault S3.
- AppMirrorRelationship (AMR) agora replica apenas PVs no estado Bound, para evitar tentativas falhas.

- Corrigido o problema em que erros eram exibidos ao obter AppVaultContent em um AppVault com um grande número de backups.
- KubeVirt VMSnapshots são excluídos das operações de restauração e failover para evitar falhas.
- Corrigido um problema no Kopia em que os snapshots estavam sendo removidos prematuramente devido à programação de retenção padrão do Kopia estar sobrescrevendo o que foi definido pelo usuário na programação.

Alterações em 25.02.1

Trident

Correções

- **Kubernetes**:
  - Corrigido um problema no trident-operator onde os nomes e versões das imagens sidecar eram preenchidos incorretamente ao usar um registro de imagens não padrão ("Problema nº 983").
  - Corrigido o problema em que as sessões multipath não conseguiam se recuperar durante um giveback do ONTAP ("Problema nº 961").

Alterações em 25.02

A partir da Trident 25.02, o resumo "Novidades" fornece detalhes sobre melhorias, correções e descontinuações para as versões do Trident e do Trident Protect.

Trident

Melhorias

- **Kubernetes**:
  - Adicionado suporte para ONTAP ASA r2 para iSCSI.
  - Adicionado suporte para desanexação forçada de volumes ONTAP-NAS durante cenários de desligamento não controlado do nó. Novos volumes ONTAP-NAS agora utilizarão políticas de exportação por volume gerenciadas pelo Trident. Fornecido um caminho de atualização para volumes existentes migrarem para o novo modelo de política de exportação ao cancelar a publicação, sem afetar as cargas de trabalho ativas.
  - Adicionada a anotação cloneFromSnapshot.
  - Adicionado suporte para clonagem de volumes entre namespaces.
  - Recursos aprimorados de autorrecuperação iSCSI para iniciar novas varreduras por host, canal, alvo e ID de LUN exatos.
  - Adicionado suporte para Kubernetes 1.32.
- **OpenShift**:
  - Adicionado suporte para preparação automática de nós iSCSI para RHCOS em clusters ROSA.
  - Adicionado suporte para OpenShift Virtualization para drivers ONTAP.
- Adicionado suporte a Fibre Channel no driver ONTAP-SAN.
- Adicionado suporte a NVMe LUKS.
- Alterado para imagem provisória para todas as imagens base.

- Adicionada descoberta e registro do estado de conexão iSCSI quando as sessões iSCSI deveriam estar conectadas, mas não estão ("Problema nº 961").
- Adicionado suporte para volumes SMB com o driver google-cloud-netapp-volumes.
- Adicionado suporte para permitir que volumes ONTAP ignorem a fila de recuperação ao serem excluídos.
- Adicionado suporte para substituir as imagens padrão usando SHAs em vez de tags.
- Adicionada a flag image-pull-secrets ao instalador tridentctl.

Correções

- **Kubernetes**:
  - Corrigidos os endereços IP de nós ausentes nas políticas de exportação automática ("Problema nº 965").
  - Corrigido o problema de políticas de exportação automáticas que mudavam prematuramente para a política por volume para ONTAP-NAS-Economy.
  - Credenciais de configuração de backend corrigidas para suportar todas as partições ARN da AWS disponíveis ("Problema #913").
  - Adicionada opção para desativar a reconciliação automática do configurador no operador Trident ("Problema nº 924").
  - Adicionado securityContext para o contêiner csi-resizer ("Problema #976").

Trident Protect

NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI.

Melhorias

- Adicionado suporte de backup e restauração para KubeVirt / OpenShift VMs de Virtualização para ambos volumeMode: File e volumeMode: Block (armazenamento de dispositivo bruto). Esse suporte é compatível com todos os drivers Trident e aprimora os recursos de proteção existentes ao replicar o armazenamento usando NetApp SnapMirror com Trident Protect.
- Adicionada a capacidade de controlar o comportamento de congelamento no nível do aplicativo para ambientes Kubevirt.
- Adicionado suporte para configurar conexões proxy do AutoSupport.
- Adicionada a capacidade de definir um segredo para criptografia de data mover (Kopia / Restic).
- Adicionada a capacidade de executar manualmente um hook de execução.
- Adicionada a capacidade de configurar restrições de contexto de segurança (SCCs) durante a instalação do Trident Protect.
- Adicionado suporte para configurar nodeSelector durante a instalação do Trident Protect.
- Adicionado suporte para proxy de saída HTTP / HTTPS para objetos AppVault.
- Ampliado ResourceFilter para permitir a exclusão de recursos com escopo de cluster.
- Adicionada compatibilidade com o token de sessão da AWS nas credenciais do S3 AppVault.
- Adicionado suporte para coleta de recursos após a execução de hooks de pré-snapshot.

Correções

- Melhoramos a gestão de volumes temporários para pular a fila de recuperação de volumes do ONTAP.
- As anotações SCC agora são restauradas aos valores originais.
- Eficiência de restauração aprimorada com suporte para operações paralelas.
- Suporte aprimorado para tempos limite de gancho de execução para aplicações maiores.

Alterações em 24.10.1

Melhorias

- **Kubernetes**: Adicionado suporte para Kubernetes 1.32.
- Adicionada descoberta e registro do estado de conexão iSCSI quando as sessões iSCSI deveriam estar conectadas, mas não estão ("Problema nº 961").

Correções

- Corrigidos os endereços IP de nós ausentes nas políticas de exportação automática ("Problema nº 965").
- Corrigido o problema de políticas de exportação automáticas que mudavam prematuramente para a política por volume para ONTAP-NAS-Economy.
- Dependências do Trident e do Trident-ASUP atualizadas para corrigir as vulnerabilidades CVE-2024-45337 e CVE-2024-45310.
- Removidos os logouts para portais não-CHAP intermitentemente não íntegros durante a autorrecuperação iSCSI ("Problema nº 961").

Alterações em 24.10

Melhorias

- O driver do Google Cloud NetApp Volumes agora está disponível para o público em geral para volumes NFS e oferece suporte ao provisionamento com reconhecimento de zonas.
- A identidade de carga de trabalho do GCP será usada como identidade de nuvem para Google Cloud NetApp Volumes com GKE.
- Adicionado `formatOptions` parâmetro de configuração aos drivers ONTAP-SAN e ONTAP-SAN-Economy para permitir que os usuários especifiquem opções de formato LUN.
- Tamanho mínimo do volume do Azure NetApp Files reduzido para 50 GiB. Novo tamanho mínimo do Azure previsto para estar geralmente disponível em novembro.
- Adicionado `denyNewVolumePools` parâmetro de configuração para restringir os drivers ONTAP-NAS-Economy e ONTAP-SAN-Economy a pools Flexvol preexistentes.
- Adicionada detecção para a inclusão, remoção ou renomeação de agregados do SVM em todos os drivers ONTAP.
- Adicionado 18 MiB de sobrecarga aos LUNs LUKS para garantir que o tamanho do PVC relatado seja utilizável.
- Melhorado o tratamento de erros de estágio e desestágio de nós ONTAP-SAN e ONTAP-SAN-Economy para permitir que o desestágio remova dispositivos após uma falha de estágio.
- Adicionado um gerador de funções personalizadas que permite aos clientes criar uma função minimalista para Trident no ONTAP.

- Adicionado registro adicional para solução de problemas `lsscsi` ("Problema nº 792").

Kubernetes

- Adicionados novos recursos do Trident para fluxos de trabalho nativos do Kubernetes:
  - Proteção de dados
  - Migração de dados
  - Recuperação de desastres
  - Mobilidade de aplicativos

  "Saiba mais sobre Trident Protect".
- Adicionada uma nova opção `--k8s-api-qps` aos instaladores para definir o valor de QPS usado pelo Trident para se comunicar com o servidor da API do Kubernetes.
- Adicionada `--node-prep`flag aos instaladores para gerenciamento automático das dependências do protocolo de storage nos nós do cluster Kubernetes. Testada e verificada a compatibilidade com Amazon Linux 2023 protocolo de storage iSCSI
- Adicionado suporte para desanexação forçada para volumes ONTAP-NAS-Economy durante cenários de desligamento não controlado do nó.
- Novos volumes NFS ONTAP-NAS-Economy usarão políticas de exportação por qtree ao usar a opção de backend `autoExportPolicy`. As qtrees só serão mapeadas para políticas de exportação restritivas de nó no momento da publicação para melhorar o controle de acesso e a segurança. As qtrees existentes serão migradas para o novo modelo de política de exportação quando Trident despublicar o volume de todos os nós, para fazer isso sem impactar as cargas de trabalho ativas.
- Adicionado suporte para Kubernetes 1.31.

Aprimoramentos experimentais

- Adicionada prévia técnica para suporte a Fibre Channel no driver ONTAP-SAN.

Correções

- **Kubernetes**:
  - Corrigido o webhook de admissão do Rancher que impedia instalações do Trident Helm ("Problema nº 839").
  - Chave de afinidade fixa nos valores do gráfico Helm ("Problema nº 898").
  - Corrigido tridentControllerPluginNodeSelector/tridentNodePluginNodeSelector não funcionará com o valor "true" ("Problema nº 899").
  - Instantâneos efêmeros excluídos criados durante a clonagem ("Problema nº 901").
- Adicionado suporte para Windows Server 2019.
- Corrigido `go mod tidy` no repositório Trident ("Problema #767").

Descontinuações

- **Kubernetes:**
  - Versão mínima do Kubernetes suportada atualizada para 1.25.
  - Removido o suporte para POD Security Policy.

Rebranding do produto

A partir da versão 24.10, Astra Trident passou a se chamar Trident (NetApp Trident). Essa mudança de nome não afeta nenhum recurso, plataforma suportada ou interoperabilidade do Trident.

Alterações em 24.06

Melhorias

- **IMPORTANTE**: O `limitVolumeSize` parâmetro agora limita os tamanhos de qtree/LUN nos drivers econômicos do ONTAP. Use o novo `limitVolumePoolSize` parâmetro para controlar os tamanhos de Flexvol nesses drivers. ("Problema nº 341").
- Adicionada a capacidade de autorrecuperação iSCSI para iniciar varreduras SCSI pelo ID exato do LUN se igroups obsoletos estiverem em uso ("Problema nº 883").
- Adicionado suporte para operações de clone de volume e redimensionamento serem permitidas mesmo quando o backend estiver em modo suspenso.
- Adicionada a capacidade de as configurações de log definidas pelo usuário para o controlador Trident serem propagadas para os pods de nó do Trident.
- Adicionado suporte no Trident para usar REST por padrão em vez de ONTAPI (ZAPI) para versões do ONTAP 9.15.1 e posteriores.
- Adicionado suporte para nomes de volumes personalizados e metadados nos backends de armazenamento ONTAP para novos volumes persistentes.
- Aprimorado o driver `azure-netapp-files` (ANF) para habilitar automaticamente o diretório de snapshots por padrão quando as opções de montagem NFS estiverem definidas para usar a versão de NFS 4.x.
- Adicionado suporte ao Bottlerocket para volumes NFS.
- Adicionado suporte de pré-visualização técnica para Google Cloud NetApp Volumes.

Kubernetes

- Adicionado suporte para Kubernetes 1.30.
- Adicionada capacidade para Trident DaemonSet limpar montarias de zumbis e arquivos residuais de rastreamento na inicialização ("Problema nº 883").
- Adicionada anotação PVC `trident.netapp.io/luksEncryption` para importação dinâmica de volumes LUKS ("Problema #849").
- Adicionada a capacidade de reconhecimento de topologia ao driver ANF.
- Adicionado suporte para nós do Windows Server 2022.

Correções

- Corrigidas falhas na instalação do Trident devido a transações obsoletas.
- Corrigido o tridentctl para ignorar mensagens de aviso do Kubernetes ("Problema #892").
- Alterada a prioridade do controlador Trident `SecurityContextConstraint` para `0` ("Problema nº 887").
- Os drivers ONTAP agora aceitam tamanhos de volume abaixo de 20 MiB ("Problema[#885").
- Correção no Trident para evitar a redução de volumes FlexVol durante a operação de redimensionamento para o driver ONTAP-SAN.

- Corrigida a falha na importação de volumes ANF com NFS v4.1.

Alterações em 24.02

Melhorias

- Adicionado suporte para Cloud Identity.
  - AKS com ANF - Azure Workload Identity será usado como identidade na nuvem.
  - EKS com FSxN - a função AWS IAM será usada como identidade na nuvem.
- Adicionado suporte para instalar Trident como um complemento no cluster EKS a partir do console do EKS.
- Adicionada a capacidade de configurar e desativar a autorrecuperação iSCSI ("Problema nº 864").
- Adicionada personalidade Amazon FSx aos drivers ONTAP para permitir a integração com AWS IAM e SecretsManager, e para permitir que Trident exclua volumes FSx com backups ("Problema #453").

Kubernetes

- Adicionado suporte para Kubernetes 1.29.

Correções

- Mensagens de aviso do ACP corrigidas, quando o ACP não está ativado ("Problema nº 866").
- Adicionado um atraso de 10 segundos antes de realizar a divisão de clones durante a exclusão de snapshot para ONTAP drivers, quando um clone está associado ao snapshot.

Descontinuações

- Removida a estrutura de atestação in-toto dos manifestos de imagem multiplataforma.

Alterações em 23.10

Correções

- Corrigida a expansão de volume se um novo tamanho solicitado for menor que o tamanho total do volume para drivers de armazenamento ontap-nas e ontap-nas-flexgroup ("Problema nº 834").
- Tamanho de volume fixo para exibir apenas o tamanho utilizável do volume durante a importação para os drivers de armazenamento ontap-nas e ontap-nas-flexgroup ("Problema nº 722").
- Correção na conversão de nomes de FlexVol para ONTAP-NAS-Economy.
- Corrigido o problema de inicialização do Trident em um nó Windows quando o nó é reinicializado.

Melhorias

Kubernetes

Adicionado suporte para Kubernetes 1.28.

Trident

- Adicionado suporte para o uso de Azure Managed Identities (AMI) com o driver de armazenamento azure-netapp-files.

- Adicionado suporte para NVMe over TCP para o driver ONTAP-SAN.
- Adicionada a capacidade de pausar o provisionamento de um volume quando o backend for definido como estado suspenso pelo usuário ("Problema nº 558").

Alterações em 23.07.1

Kubernetes: Corrigida a exclusão do daemonset para suportar atualizações sem tempo de inatividade ("Problema nº 740").

Alterações em 23.07

Correções

Kubernetes

- Correção na atualização do Trident para ignorar pods antigos presos no estado de término ("Problema nº 740").
- Adicionada tolerância à definição "transient-trident-version-pod" ("Problema #795").

Trident

- Corrigidas as solicitações ONTAPI (ZAPI) para garantir que os números de série do LUN sejam consultados ao obter os atributos do LUN, a fim de identificar e corrigir dispositivos iSCSI fantasmas durante as operações de preparação de nós.
- Correção no tratamento de erros no código do driver de armazenamento ("Problema nº 816").
- Correção no redimensionamento de cotas ao usar drivers ONTAP com use-rest=true.
- Correção na criação de clone de LUN em ontap-san-economy.
- Reverter o campo de informações de publicação de `rawDevicePath` para `devicePath`; adicionada lógica para preencher e recuperar (em alguns casos) `devicePath` campo.

Melhorias

Kubernetes

- Adicionado suporte para importação de snapshots pré-provisionados.
- Implantação minimizada e permissões do daemonset linux ("Problema nº 817").

Trident

- Não é mais exibido o campo de estado para volumes e snapshots "online".
- Atualiza o estado do backend se o ONTAP backend estiver offline ("Problemas #801", "#543").
- O número de série do LUN é sempre recuperado e publicado durante o ControllerVolumePublish workflow.
- Adicionada lógica adicional para verificar o número de série e o tamanho do dispositivo iSCSI multipath.
- Verificação adicional para volumes iSCSI para garantir que o dispositivo multipath correto esteja descompactado.

Aprimoramento experimental

Adicionado suporte em prévia técnica para NVMe over TCP para o driver ONTAP-SAN.

Documentação

Muitas melhorias de organização e formatação foram feitas.

Descontinuações

Kubernetes

- Removido o suporte para snapshots v1beta1.
- Removido o suporte para volumes e classes de armazenamento pré-CSI.
- Versão mínima do Kubernetes suportada atualizada para 1.22.

Alterações em 23.04

> O recurso de desanexação forçada de volumes ONTAP-SAN-* é compatível apenas com versões do Kubernetes que possuem o recurso de desligamento não gradual do nó ativado. A desanexação forçada deve ser habilitada durante a instalação usando a `--enable-force-detach` flag do instalador Trident.

Correções

- Corrigido o Trident Operator para usar IPv6 localhost para instalação quando especificado em spec.
- Corrigidas as permissões da função de cluster do operador Trident para estarem sincronizadas com as permissões do pacote ("Problema nº 799").
- Corrigido problema com a anexação de volume de bloco bruto em vários nós no modo RWX.
- Correção do suporte à clonagem de FlexGroup e importação de volumes SMB.
- Corrigido problema em que o controlador Trident não conseguia ser desligado imediatamente ("Problema nº 811").
- Adicionada correção para listar todos os nomes de igroup associados a uma LUN especificada provisionada com drivers ontap-san-*.
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Erro de compilação corrigido para a arquitetura s390 ("Problema #537").
- Corrigido o nível de registro incorreto durante as operações de montagem de volume ("Problema #781").
- Corrigido erro potencial de asserção de tipo ("Problema #802").

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1.27.
  - Adicionado suporte para importação de volumes LUKS.
  - Adicionada compatibilidade com o modo de acesso ReadWriteOncePod em PVC.
  - Adicionado suporte para desanexação forçada para volumes ONTAP-SAN-* durante cenários de desligamento não controlado do nó.

    - Todos os volumes ONTAP-SAN-* agora usarão igroups por nó. Os LUNs serão mapeados para igroups somente enquanto estiverem ativamente publicados nesses nós para melhorar nossa postura de segurança. Os volumes existentes serão migrados para o novo esquema de igroups de forma oportunista quando Trident determinar que é seguro fazê-lo sem impactar as cargas de trabalho ativas ("Problema nº 758").
    - Segurança do Trident aprimorada ao limpar igroups gerenciados pelo Trident e não utilizados dos backends ONTAP-SAN-*.
- Adicionado suporte para volumes SMB com Amazon FSx aos drivers de armazenamento ontap-nas-economy e ontap-nas-flexgroup.
- Adicionado suporte para compartilhamentos SMB com os drivers de armazenamento ontap-nas, ontap-nas-economy e ontap-nas-flexgroup.
- Adicionado suporte para nós arm64 ("Problema nº 732").
- Procedimento de desligamento do Trident aprimorado, desativando primeiro os servidores de API ("Problema nº 811").
- Adicionada compatibilidade com compilação multiplataforma para hosts Windows e arm64 ao Makefile; consulte BUILD.md.

Descontinuações

Kubernetes: Grupos de interfaces com escopo de backend não serão mais criados ao configurar os drivers ontap-san e ontap-san-economy ("Problema nº 758").

Alterações em 23.01.1

Correções

- Corrigido o Trident Operator para usar IPv6 localhost para instalação quando especificado em spec.
- Corrigidas as permissões da função de cluster do operador Trident para estarem sincronizadas com as permissões do bundle "Problema nº 799".
- Adicionada uma correção para permitir que processos externos sejam executados até a conclusão.
- Corrigido problema com a anexação de volume de bloco bruto em vários nós no modo RWX.
- Correção do suporte à clonagem de FlexGroup e importação de volumes SMB.

Alterações em 23.01

Kubernetes 1.27 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.

Correções

- Kubernetes: adicionadas opções para excluir a criação de Pod Security Policy para corrigir instalações do Trident via Helm ("Problemas nº 783, nº 794").

Melhorias

Kubernetes

- Adicionado suporte para Kubernetes 1.26.

- Melhoria geral da utilização de recursos do Trident ("Problema nº 757").
- Adicionada automação para detectar e corrigir sessões iSCSI interrompidas ou obsoletas em nós de host.
- Adicionado suporte para expandir volumes criptografados com LUKS.
- Kubernetes: Adicionado suporte à rotação de credenciais para volumes criptografados com LUKS.

Trident

- Adicionado suporte para volumes SMB com Amazon FSx for NetApp ONTAP ao driver de armazenamento ontap-nas.
- Adicionado suporte para permissões NTFS ao usar volumes SMB.
- Adicionado suporte para pools de armazenamento para volumes do GCP com nível de serviço CVS.
- Adicionado suporte ao uso opcional de flexgroupAggregateList ao criar FlexGroups com o driver de armazenamento ontap-nas-flexgroup.
- Desempenho aprimorado para o driver de armazenamento ontap-nas-economy ao gerenciar vários volumes FlexVol
- Ativadas as atualizações do dataLIF para todos os drivers de armazenamento ONTAP NAS.
- Atualizada a convenção de nomenclatura de implantação do Trident e DaemonSet para refletir o sistema operacional do nó host.

Descontinuações

- Kubernetes: versão mínima do Kubernetes suportada atualizada para 1.21.
- Não é mais necessário especificar DataLIFs ao configurar `ontap-san` ou `ontap-san-economy`drivers.

Alterações em 22.10

Você deve ler as seguintes informações críticas antes de atualizar para Trident 22.10.

Informações críticas sobre Trident 22.10

- Kubernetes 1.25 agora é compatível com Trident. Você deve atualizar o Trident para 22.10 antes de atualizar para o Kubernetes 1.25.
- Trident agora impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Correções

- Corrigido problema específico do backend ONTAP criado usando `credentials` campo que não entrava em funcionamento durante a atualização 22.07.0 ("Problema #759").
- **Docker:** Corrigido um problema que impedia o plugin de volume do Docker de iniciar em alguns ambientes ("Problema #548" e "Problema #760").
- Corrigido problema de SLM específico para backends SAN ONTAP para garantir que apenas um subconjunto de dataLIFs pertencentes a nós de relatório sejam publicados.

- Corrigido problema de desempenho em que varreduras desnecessárias de LUNs iSCSI ocorriam ao conectar um volume.
- Removidas as tentativas granulares dentro do fluxo de trabalho iSCSI do Trident para falhar rapidamente e reduzir os intervalos de repetição externa.
- Corrigido um problema em que um erro era retornado ao limpar um dispositivo iSCSI quando o dispositivo multipath correspondente já havia sido limpo.

Melhorias

- Kubernetes:
  - Adicionado suporte para Kubernetes 1.25. Você deve atualizar Trident para 22.10 antes de atualizar para Kubernetes 1.25.
  - Adicionado um ServiceAccount, ClusterRole e ClusterRoleBinding separados para a implantação do Trident e DaemonSet para permitir futuras melhorias de permissões.
  - Adicionado suporte para "compartilhamento de volumes entre namespaces".
- Todos os drivers de armazenamento Trident `ontap-*` agora funcionam com a API REST do ONTAP.
- Adicionado novo operador yaml (`bundle_post_1_25.yaml`) sem um `PodSecurityPolicy` para suportar Kubernetes 1.25.
- Adicionado "suporte para volumes criptografados com LUKS" para `ontap-san` e `ontap-san-economy` drivers de armazenamento.
- Adicionado suporte para nós do Windows Server 2019.
- Adicionado "Suporte para volumes SMB em nós Windows" através do `azure-netapp-files` driver de armazenamento.
- A detecção automática de switchover MetroCluster para drivers ONTAP agora está geralmente disponível.

Descontinuações

- **Kubernetes:** versão mínima do Kubernetes suportada atualizada para 1.20.
- Driver Astra Data Store (ADS) removido.
- Removido suporte para `yes` e `smart` opções para `find_multipaths` ao configurar multipath de nó de trabalho para iSCSI.

Alterações em 22.07

Correções

Kubernetes

- Corrigido problema no tratamento de valores booleanos e numéricos para o seletor de nós ao configurar Trident com Helm ou Trident Operator. ("GitHub issue nº 700")
- Corrigido problema no tratamento de erros em caminhos não-CHAP, de forma que o kubelet tente novamente em caso de falha. "GitHub issue nº 736")

Melhorias

- Transição de k8s.gcr.io para registry.k8s.io como registro padrão para imagens CSI

- Os volumes ONTAP-SAN agora usarão igroups por nó e mapearão LUNs para igroups somente enquanto estiverem ativamente publicados nesses nós para aprimorar nossa postura de segurança. Os volumes existentes serão migrados para o novo esquema de igroups de forma oportunista quando o Trident determinar que é seguro fazê-lo sem impactar as cargas de trabalho ativas.
- Incluído um ResourceQuota nas instalações do Trident para garantir que o Trident DaemonSet seja agendado quando o consumo de PriorityClass for limitado por padrão.
- Adicionado suporte para Recursos de Rede ao driver do Azure NetApp Files. ("GitHub issue nº 717")
- Adicionada detecção automática de switchover MetroCluster (tech preview) aos drivers ONTAP. ("GitHub issue nº 228")

Descontinuações

- **Kubernetes:** versão mínima do Kubernetes suportada atualizada para 1.19.
- A configuração do backend não permite mais múltiplos tipos de autenticação em uma única configuração.

Remoções

- O driver AWS CVS (obsoleto desde 22.04) foi removido.
- Kubernetes
  - Removida a capacidade SYS_ADMIN desnecessária dos pods dos nós.
  - Reduz o nodeprep a informações simples do host e descoberta ativa de serviços para realizar uma confirmação da melhor forma possível de que os serviços NFS/iSCSI estão disponíveis nos nós de trabalho.

Documentação

Uma nova "Padrões de segurança do Pod" seção (PSS) foi adicionada detalhando as permissões habilitadas pelo Trident na instalação.

Alterações em 22.04

NetApp está continuamente aprimorando e melhorando seus produtos e serviços. Aqui estão alguns dos recursos mais recentes em Trident. Para versões anteriores, consulte "Versões anteriores da documentação".

> Se você estiver atualizando de qualquer versão anterior do Trident e usar o Azure NetApp Files, o `location` parâmetro de configuração agora é um campo obrigatório e único.

Correções

- Melhoria na análise dos nomes dos iniciadores iSCSI. ("GitHub issue nº 681")
- Corrigido problema em que os parâmetros da classe de armazenamento CSI não eram permitidos. ("Issue do GitHub #598")
- Correção de declaração de chave duplicada no Trident CRD. ("Problema do GitHub #671")
- Corrigidos os registros de Snapshot CSI imprecisos. ("Problema do GitHub #629"))
- Problema corrigido com a remoção da publicação de volumes em nós excluídos. ("GitHub issue nº 691")
- Adicionada a funcionalidade de tratamento de inconsistências do sistema de arquivos em dispositivos de bloco. ("Problema do GitHub #656")

- Corrigido problema ao obter imagens de auto-support ao definir o `imageRegistry` flag durante a instalação. ("Problema do GitHub #715")
- Corrigido um problema em que o driver do Azure NetApp Files não conseguia clonar um volume com várias regras de exportação.

Melhorias

- As conexões de entrada para os endpoints seguros do Trident agora exigem no mínimo TLS 1.3. ("Problema do GitHub #698")
- Trident agora adiciona cabeçalhos HSTS às respostas de seus endpoints seguros.
- Trident agora tenta habilitar automaticamente o recurso de permissões unix do Azure NetApp Files.
- **Kubernetes**: O daemonset Trident agora é executado na classe de prioridade system-node-critical. ("Problema do GitHub #694")

Remoções

O driver E-Series (desativado desde 20.07) foi removido.

Alterações em 22.01.1

Correções

- Problema corrigido com a remoção da publicação de volumes em nós excluídos. ("GitHub issue nº 691")
- Corrigido o erro que causava pânico ao acessar campos nulos para espaço agregado nas respostas da API ONTAP.

Alterações em 22.01.0

Correções

- **Kubernetes:** Aumente o tempo de repetição de backoff do registro de nós para clusters grandes.
- Corrigido um problema em que o driver azure-netapp-files podia ser confundido por múltiplos recursos com o mesmo nome.
- Agora, os DataLIFs IPv6 do SAN ONTAP funcionam se especificados entre colchetes.
- Corrigido problema em que a tentativa de importar um volume já importado retornava EOF, deixando o PVC em estado pendente. ("Problema do GitHub #489")
- Corrigido problema em que o desempenho do Trident ficava mais lento quando mais de 32 snapshots eram criados em um SolidFire volume.
- Substituído SHA-1 por SHA-256 na criação do certificado SSL.
- Corrigido o driver do Azure NetApp Files para permitir nomes de recursos duplicados e limitar as operações a um único local.
- Corrigido o driver do Azure NetApp Files para permitir nomes de recursos duplicados e limitar as operações a um único local.

Melhorias

- Aprimoramentos do Kubernetes:

- Adicionado suporte para Kubernetes 1.23.
- Adicionar opções de agendamento para Trident pods quando instalados via Trident Operator ou Helm. ("GitHub issue nº 651")

- Permitir volumes entre regiões no driver do GCP. ("GitHub issue nº 633")
- Adicionado suporte para a opção 'unixPermissions' aos volumes do Azure NetApp Files. ("GitHub issue nº 666")

Descontinuações

A interface REST do Trident só pode escutar e atender em endereços 127.0.0.1 ou [::1]

Alterações em 21.10.1

> A versão v21.10.0 apresenta um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado novamente ao cluster Kubernetes. Esse problema foi corrigido na v21.10.1 (GitHub issue 669).

Correções

- Corrigida uma possível condição de corrida ao importar um volume em um backend CVS do GCP, resultando em falha na importação.
- Corrigido um problema que pode colocar o controlador Trident em um estado CrashLoopBackOff quando um nó é removido e depois adicionado novamente ao cluster Kubernetes (GitHub issue 669).
- Corrigido o problema em que as SVMs não eram mais detectadas se nenhum nome de SVM fosse especificado (GitHub issue 612).

Alterações em 21.10.0

Correções

- Corrigido o problema em que clones de volumes XFS não podiam ser montados no mesmo nó que o volume de origem (GitHub issue 514).
- Corrigido o problema em que Trident registrava um erro fatal ao ser desligado (GitHub issue 597).
- Correções relacionadas a Kubernetes:
  - Retorne o espaço usado de um volume como o mínimo restoreSize ao criar snapshots com `ontap-nas` e `ontap-nas-flexgroup` drivers (GitHub issue 645).
  - Corrigido o problema em que `Failed to expand filesystem` erro era registrado após o redimensionamento do volume (GitHub issue 560).
  - Corrigido problema em que um pod podia ficar preso em `Terminating` state (GitHub issue 572).
  - Corrigido o caso em que um `ontap-san-economy` FlexVol poderia estar cheio de LUNs de snapshot (GitHub issue 533).
  - Corrigido problema com instalador YAML personalizado com imagem diferente (GitHub issue 613).
  - Cálculo corrigido do tamanho do instantâneo (GitHub issue 611).
  - Corrigido o problema em que todos os instaladores do Trident podiam identificar o Kubernetes puro como OpenShift (GitHub issue 639).

- Corrigido o operador Trident para interromper a reconciliação se o servidor da API do Kubernetes estiver inacessível (GitHub issue 599).

Melhorias

- Adicionado suporte para a opção `unixPermissions` em volumes de desempenho do GCP-CVS.
- Adicionado suporte para volumes CVS otimizados para escala no GCP na faixa de 600 GiB a 1 TiB.
- Melhorias relacionadas ao Kubernetes:
  - Adicionado suporte para Kubernetes 1.22.
  - Habilitamos o operador Trident e o gráfico Helm para funcionar com o Kubernetes 1.22 (GitHub issue 628).
  - Adicionada imagem do operador ao `tridentctl` comando de imagens (GitHub issue 570).

Aprimoramentos experimentais

- Adicionado suporte para replicação de volumes no driver `ontap-san`.
- Adicionado suporte **prévia técnica** REST para os `ontap-nas-flexgroup`, `ontap-san`, e `ontap-nas-economy` drivers.

Problemas conhecidos

Problemas conhecidos identificam problemas que podem impedi-lo de usar o produto com sucesso.

- Ao atualizar um cluster Kubernetes da versão 1.24 para a 1.25 ou posterior que tenha Trident instalado, você deve atualizar o arquivo values.yaml para definir `excludePodSecurityPolicy` para `true` ou adicionar `--set excludePodSecurityPolicy=true` ao comando `helm upgrade` antes de poder atualizar o cluster.
- Trident agora impõe um valor em branco `fsType` (`fsType=""` para volumes que não possuem o `fsType` especificado em seu StorageClass. Ao trabalhar com Kubernetes 1.17 ou posterior, Trident oferece suporte ao fornecimento de um valor em branco `fsType` para volumes NFS. Para volumes iSCSI, é necessário definir o `fsType` em seu StorageClass ao impor um `fsGroup` usando um contexto de segurança.
- Ao usar um backend em várias instâncias do Trident, cada arquivo de configuração de backend deve ter um valor `storagePrefix` diferente para backends ONTAP ou usar um valor `TenantName` diferente para backends SolidFire. O Trident não consegue detectar volumes que outras instâncias do Trident criaram. Tentar criar um volume existente em backends ONTAP ou SolidFire é bem-sucedido, porque o Trident trata a criação de volume como uma operação idempotente. Se `storagePrefix` ou `TenantName` não forem diferentes, pode haver colisão de nomes para volumes criados no mesmo backend.
- Ao instalar Trident (usando `tridentctl` ou o Trident Operator) e usando `tridentctl` para gerenciar o Trident, você deve garantir que a variável de ambiente `KUBECONFIG` esteja configurada. Isso é necessário para indicar o cluster Kubernetes que `tridentctl` deve ser utilizado. Ao trabalhar com múltiplos ambientes Kubernetes, você deve garantir que o arquivo `KUBECONFIG` seja referenciado corretamente.
- Para realizar exigência de espaço online para iSCSI PVs, o sistema operacional subjacente no nó de trabalho pode exigir que opções de montagem sejam passadas para o volume. Isso é verdade para instâncias RHEL/Red Hat Enterprise Linux CoreOS (RHCOS), que exigem o `discard` "opção de montagem"; certifique-se de que o descarte mountOption esteja incluído em seu[`StorageClass`^] para suportar o descarte de blocos online.

- Se você tiver mais de uma instância do Trident por cluster Kubernetes, Trident não poderá se comunicar com outras instâncias nem descobrir outros volumes que elas criaram, o que leva a comportamentos inesperados e incorretos se mais de uma instância estiver em execução em um cluster. Deve haver apenas uma instância do Trident por cluster Kubernetes.
- Se objetos baseados em Trident `StorageClass` forem excluídos do Kubernetes enquanto Trident estiver offline, Trident não remove as classes de armazenamento correspondentes do seu banco de dados quando voltar a ficar online. Você deve excluir essas classes de armazenamento usando `tridentctl` ou a API REST.
- Se um usuário excluir um PV provisionado pelo Trident antes de excluir o PVC correspondente, Trident não exclui automaticamente o volume subjacente. Você deve remover o volume via `tridentctl` ou a API REST.
- ONTAP não pode provisionar mais de um FlexGroup por vez simultaneamente, a menos que o conjunto de agregados seja exclusivo para cada solicitação de provisionamento.
- Ao usar Trident sobre IPv6, você deve especificar `managementLIF` e `dataLIF` na definição do backend entre colchetes. Por exemplo, `[fd20:8b1e:b258:2000:f816:3eff:feec:0]`.

> Não é possível especificar `dataLIF` em um backend SAN ONTAP. Trident descobre todas as LIFs iSCSI disponíveis e as utiliza para estabelecer a sessão multipath.

- Se estiver usando o driver `solidfire-san` com OpenShift 4.5, certifique-se de que os nós de trabalho subjacentes usem MD5 como algoritmo de autenticação CHAP. Algoritmos CHAP seguros compatíveis com FIPS, SHA1, SHA-256 e SHA3-256 estão disponíveis com o Element 12.7.

Encontre mais informações

- "Trident GitHub"
- "Blogs do Trident"

Versões anteriores da documentação

Se você não estiver executando Trident 25.10, a documentação para versões anteriores está disponível com base no "Ciclo de vida do Trident".

- "Trident 25.06"
- "Trident 25.02"
- "Trident 24.10"
- "Trident 24.06"
- "Trident 24.02"
- "Trident 23.10"
- "Trident 23.07"
- "Trident 23.04"
- "Trident 23.01"

NetApp Trident suporte para sistemas de storage ONTAP ASA r2

NetApp Trident 25.02 e versões posteriores suportam sistemas NetApp ASA r2 como backends de armazenamento. Consulte "Sistemas ASA r2" para mais informações.

ASA r2 sistemas requerem o `ontap-san`driver. Trident não oferece suporte ao `ontap-san-economy`driver para sistemas ASA r2.

Ao especificar `ontap-san` como o `storageDriverName` na configuração do backend, o Trident detecta automaticamente o sistema de storage ASA r2.

Trident oferece proteção de dados limitada para sistemas ASA r2 com Trident protect.

Os protocolos SAN suportados dependem da sua versão do Trident:

- Trident 25.02 e versões posteriores suportam iSCSI.
- Trident 25.06 e versões posteriores suportam NVMe/TCP além de iSCSI.

Você deve atribuir pelo menos um agregado à máquina virtual de storage (SVM) para o storage de backend ONTAP. Consulte "Atribuindo agregados a SVMs em sistemas ASA r2" para obter instruções.

Operações suportadas

- Provisionamento de volumes persistentes (PVs)
- Provisionamento dinâmico de volumes
- Criando e excluindo volumes
- Clonando volumes
- Expansão de volumes
- Gerenciando classes de armazenamento

Operações não suportadas

- LUKS criptografia
- SnapMirror volume replicação
- Limitação do uso agregado
- Modo de reserva de espaço
- Instantâneos
- Camadas

Para obter mais informações, consulte "Opções e exemplos de configuração do ONTAP SAN" .

Problemas conhecidos

Os problemas conhecidos identificam problemas que podem impedir você de usar esta versão do produto com sucesso.

Os seguintes problemas conhecidos afetam a versão atual:

A restauração de backups Restic de arquivos grandes pode falhar

Ao restaurar arquivos de 30GB ou maiores de um backup do Amazon S3 feito usando Restic, a operação de restauração pode falhar. Como solução alternativa, faça backup dos dados usando Kopia como o data mover (Kopia é o data mover padrão para backups). Consulte "Proteja aplicativos usando Trident Protect" para instruções.

Comece agora

Saiba mais sobre Trident

Saiba mais sobre Trident

Trident é um projeto de código aberto com suporte completo, mantido por NetApp. Ele foi projetado para ajudar você a atender às demandas de persistência de sua aplicação em contêineres, utilizando interfaces padrão do setor, como a Container Storage Interface (CSI).

O que é Trident?

Netapp Trident permite o consumo e o gerenciamento de recursos de storage em todas as plataformas de storage NetApp populares, na nuvem pública ou em ambientes locais, incluindo clusters ONTAP locais (AFF, FAS e ASA), ONTAP Select, Cloud Volumes ONTAP, Element software (NetApp HCI, SolidFire), Azure NetApp Files e Amazon FSx for NetApp ONTAP.

Trident é um orquestrador de storage dinâmico compatível com a Container Storage Interface (CSI) que se integra nativamente com "Kubernetes". Trident é executado como um único Controller Pod mais um Node Pod em cada nó de trabalho no cluster. Consulte "Arquitetura do Trident" para obter detalhes.

Trident também oferece integração direta com o ecossistema Docker para plataformas de storage NetApp. O NetApp Docker Volume Plugin (nDVP) suporta o provisionamento e o gerenciamento de recursos de storage da plataforma de storage para hosts Docker. Consulte "Implantar Trident para Docker" para obter detalhes.

> ⓘ Se esta for a sua primeira vez usando o Kubernetes, você deve se familiarizar com o "Conceitos e ferramentas do Kubernetes".

Integração do Kubernetes com produtos NetApp

O portfólio de produtos de storage da NetApp integra-se a muitos aspectos de um cluster Kubernetes, fornecendo recursos avançados de gerenciamento de dados que aprimoram a funcionalidade, a capacidade, o desempenho e a disponibilidade da implantação do Kubernetes.

Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" é um serviço AWS totalmente gerenciado que permite iniciar e executar sistemas de arquivos com tecnologia do NetApp ONTAP sistema operacional de storage.

Azure NetApp Files

"Azure NetApp Files" é um serviço de compartilhamento de arquivos do Azure de nível empresarial, desenvolvido pela NetApp. Você pode executar suas cargas de trabalho baseadas em arquivos mais exigentes nativamente no Azure, com o desempenho e o gerenciamento de rich data que você espera da NetApp.

Cloud Volumes ONTAP

"Cloud Volumes ONTAP" é um dispositivo de storage somente de software que executa o software de gerenciamento de dados ONTAP na nuvem.

Google Cloud NetApp Volumes

"Google Cloud NetApp Volumes" é um serviço de storage de arquivos totalmente gerenciado no Google Cloud que oferece storage de arquivos de alto desempenho e nível empresarial.

Software Element

"Elemento" permite ao administrador de storage consolidar cargas de trabalho, garantindo desempenho e possibilitando uma infraestrutura de storage simplificada e otimizada.

NetApp HCI

"NetApp HCI" simplifica a gestão e a escalabilidade do centro de dados, automatizando tarefas rotineiras e permitindo que os administradores de infraestrutura se concentrem em funções mais importantes.

Trident pode provisionar e gerenciar dispositivos de storage para aplicativos em contêineres diretamente na plataforma de storage NetApp HCI.

NetApp ONTAP

"NetApp ONTAP" é o sistema operacional de storage unificado multiprotocolo da NetApp que oferece recursos avançados de gerenciamento de dados para qualquer aplicação.

Os sistemas ONTAP possuem configurações all-flash, híbridas ou totalmente em HDD e oferecem diversos modelos de implantação: clusters FAS, AFA e ASA locais, ONTAP Select e Cloud Volumes ONTAP. Trident oferece suporte a esses modelos de implantação do ONTAP.

Arquitetura do Trident

Trident é executado como um único Controller Pod mais um Node Pod em cada nó de trabalho do cluster. O Node Pod deve estar em execução em qualquer host onde você deseja montar um volume Trident.

Entendendo os pods do controlador e os pods do nó

Trident é implantado como um único Pod do Trident Controller e um ou mais Pods de nó do Trident no cluster Kubernetes e usa os contêineres sidecar CSI padrão do Kubernetes para simplificar a implantação dos plugins CSI. "Contêineres Sidecar CSI do Kubernetes" são mantidos pela comunidade de armazenamento do Kubernetes.

Kubernetes "seletores de nós" e "tolerations e taints" são usados para restringir um pod a ser executado em um nó específico ou preferencial. Você pode configurar seletores de nós e tolerâncias para pods de controlador e de nó durante a instalação do Trident.

- O plugin do controlador lida com o provisionamento e gerenciamento de volumes, como snapshots e redimensionamento.
- O plugin do nó gerencia a conexão do storage ao nó.

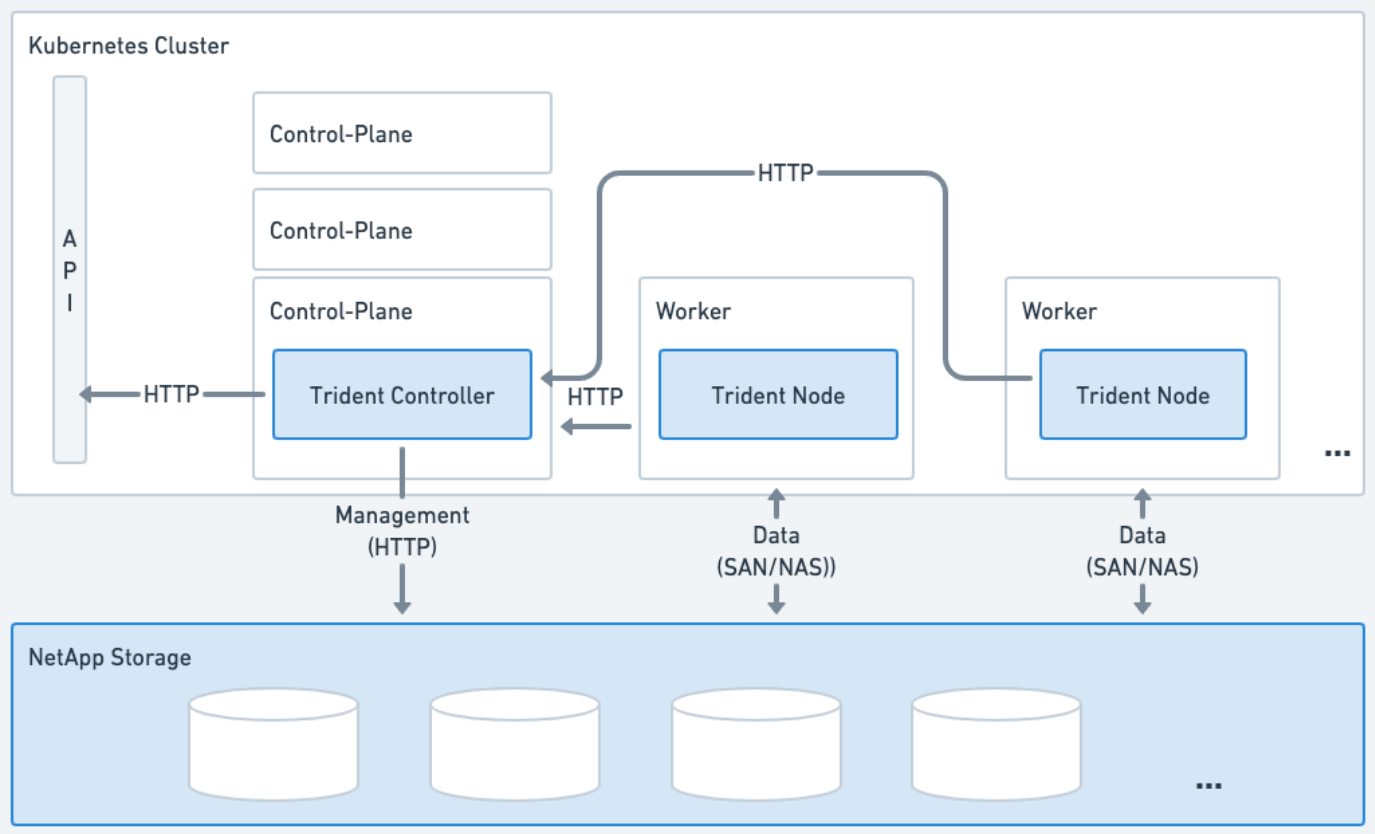


Figura 1. Trident implantado no cluster Kubernetes

Pod do Trident Controller

O Trident Controller Pod é um único Pod que executa o plugin CSI Controller.

- Responsável pelo provisionamento e gerenciamento de volumes em NetApp storage
- Gerenciado por uma implantação do Kubernetes
- Pode ser executado no plano de controle ou nos nós de trabalho, dependendo dos parâmetros de instalação.

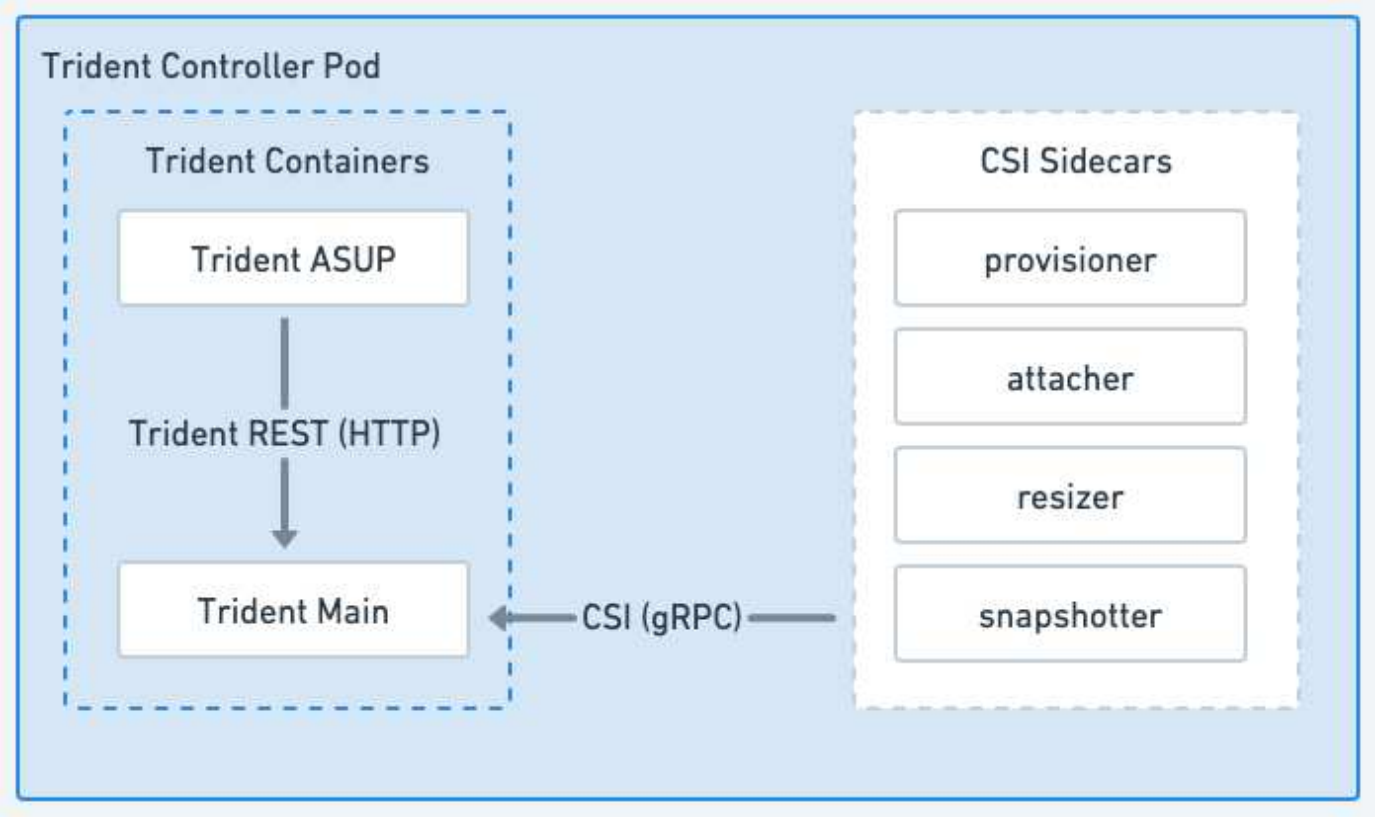


Figura 2. Diagrama do Pod do Controlador Trident

Pods de nó do Trident

Os Pods do Trident Node são Pods privilegiados que executam o plugin CSI Node.

- Responsável por montar e desmontar o storage para os Pods em execução no host
- Gerenciado por um Kubernetes DaemonSet
- Deve ser executado em qualquer nó que monte NetApp storage

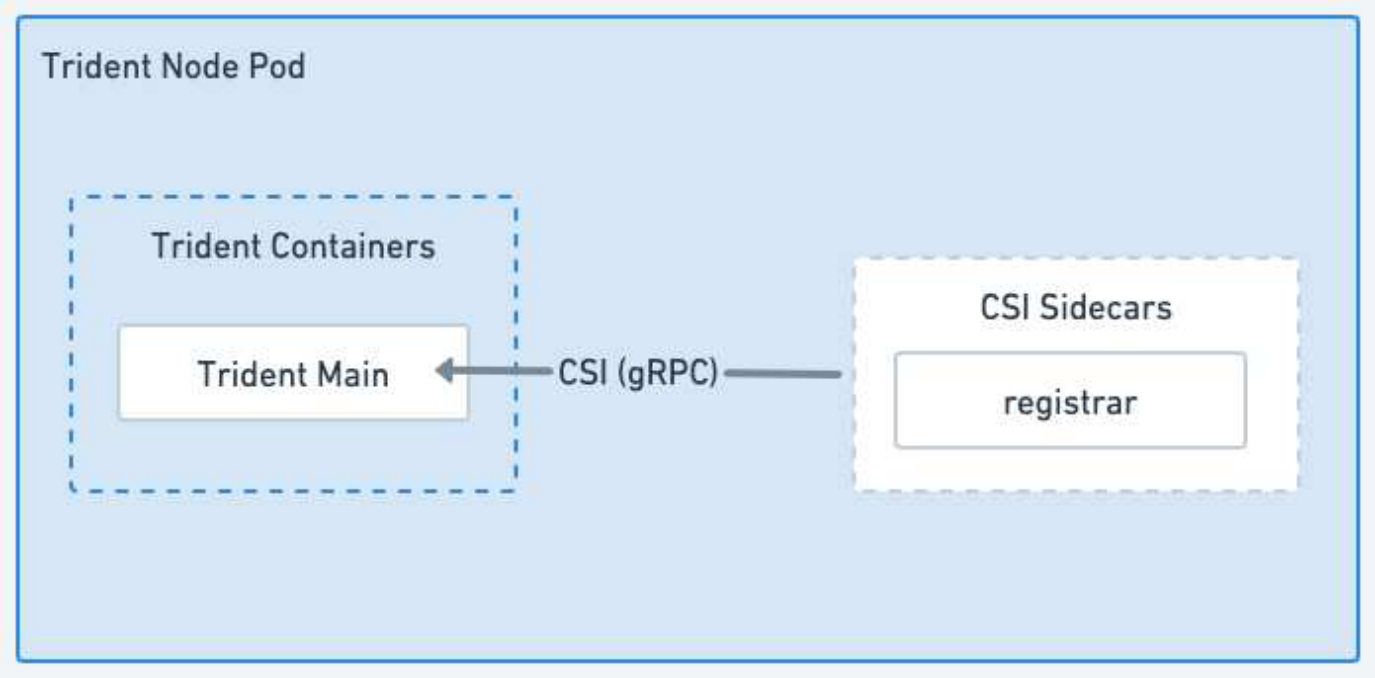


Figura 3. Diagrama do Trident Node Pod

Arquiteturas de cluster Kubernetes suportadas

Trident é compatível com as seguintes arquiteturas do Kubernetes:

Arquiteturas de cluster Kubernetes	Suportado	Instalação padrão
Mestre único, compute	Sim	Sim
Múltiplos mestres, compute	Sim	Sim
Mestre, `etcd`, computacional	Sim	Sim
Mestre, infraestrutura, compute	Sim	Sim

Conceitos

Provisionamento

O provisionamento no Trident possui duas fases principais. A primeira fase associa uma classe de armazenamento ao conjunto de pools de storage de backend adequados e ocorre como uma preparação necessária antes do provisionamento. A segunda fase inclui a criação do volume em si e requer a escolha de um pool de storage dentre aqueles associados à classe de armazenamento do volume em questão.

Associação de classe de storage

A associação de pools de storage de backend a uma classe de armazenamento depende tanto dos atributos solicitados pela classe de armazenamento quanto de suas `storagePools`, `additionalStoragePools` e `excludeStoragePools` listas. Ao criar uma classe de armazenamento, Trident compara os atributos e pools

oferecidos por cada um de seus backends com aqueles solicitados pela classe de armazenamento. Se os atributos e o nome de um pool de storage corresponderem a todos os atributos e nomes de pool solicitados, Trident adiciona esse pool de storage ao conjunto de pools de storage adequados para essa classe de armazenamento. Além disso, Trident adiciona todos os pools de storage listados na `additionalStoragePools` lista a esse conjunto, mesmo que seus atributos não atendam a todos ou a qualquer um dos atributos solicitados pela classe de armazenamento. Você deve usar a `excludeStoragePools` lista para substituir e remover pools de storage do uso para uma classe de armazenamento. Trident executa um processo semelhante sempre que você adiciona um novo backend, verificando se seus pools de storage atendem aos das classes de armazenamento existentes e removendo quaisquer pools que tenham sido marcados como excluídos.

Criação de volume

Trident utiliza as associações entre classes de armazenamento e pools de armazenamento para determinar onde provisionar os volumes. Ao criar um volume, Trident primeiro obtém o conjunto de pools de armazenamento para a classe de armazenamento desse volume e, se você especificar um protocolo para o volume, Trident remove aqueles pools de armazenamento que não podem fornecer o protocolo solicitado (por exemplo, um backend NetApp HCI/SolidFire não pode fornecer um volume baseado em arquivos, enquanto um backend ONTAP NAS não pode fornecer um volume baseado em blocos). Trident randomiza a ordem desse conjunto resultante para facilitar uma distribuição uniforme de volumes e, em seguida, itera sobre ele, tentando provisionar o volume em cada pool de armazenamento, um de cada vez. Se tiver sucesso em um deles, retorna sucesso, registrando quaisquer falhas encontradas no processo. Trident retorna uma falha **somente se** não conseguir provisionar em **todos** os pools de armazenamento disponíveis para a classe de armazenamento e o protocolo solicitados.

Instantâneos de volume

Saiba mais sobre como Trident lida com a criação de snapshots de volume para seus drivers.

Saiba mais sobre a criação de snapshots de volume

- Para os `ontap-nas`, `ontap-san`, e `azure-netapp-files` drivers, cada Persistent Volume (PV) é mapeado para um FlexVol volume. Como resultado, snapshots de volume são criados como snapshots NetApp. A tecnologia de snapshot NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot concorrentes. Essas cópias de snapshot são extremamente eficientes tanto no tempo necessário para criá-las quanto no espaço de storage.
- Para o `ontap-nas-flexgroup` driver, cada Persistent Volume (PV) é mapeado para um FlexGroup. Como resultado, snapshots de volume são criados como snapshots de NetApp FlexGroup. A tecnologia de snapshot da NetApp oferece mais estabilidade, escalabilidade, capacidade de recuperação e desempenho do que as tecnologias de snapshot concorrentes. Essas cópias de snapshot são extremamente eficientes tanto no tempo necessário para criá-las quanto no espaço de storage.
- Para o `ontap-san-economy` driver, PVs são mapeados para LUNs criados em volumes FlexVol compartilhados. VolumeSnapshots de PVs são obtidos realizando FlexClones do LUN associado. A tecnologia ONTAP FlexClone possibilita criar cópias até mesmo dos maiores conjuntos de dados quase instantaneamente. As cópias compartilham blocos de dados com seus originais, consumindo armazenamento apenas para os metadados necessários.
- Para o `solidfire-san` driver, cada PV corresponde a um LUN criado no software NetApp Element/NetApp HCI cluster. VolumeSnapshots são representados por snapshots do Element do LUN subjacente. Esses snapshots são cópias de ponto no tempo e ocupam apenas uma pequena quantidade de recursos e espaço do sistema.
- Ao trabalhar com os `ontap-nas` e `ontap-san` drivers, os snapshots ONTAP são cópias de ponto no

tempo do FlexVol e consomem espaço no próprio FlexVol. Isso pode resultar na redução da quantidade de espaço gravável no volume ao longo do tempo, à medida que snapshots são criados/agendados. Uma maneira simples de resolver isso é aumentar o volume redimensionando-o pelo Kubernetes. Outra opção é excluir snapshots que não são mais necessários. Quando um VolumeSnapshot criado pelo Kubernetes é excluído, o Trident excluirá o snapshot ONTAP associado. Snapshots ONTAP que não foram criados pelo Kubernetes também podem ser excluídos.

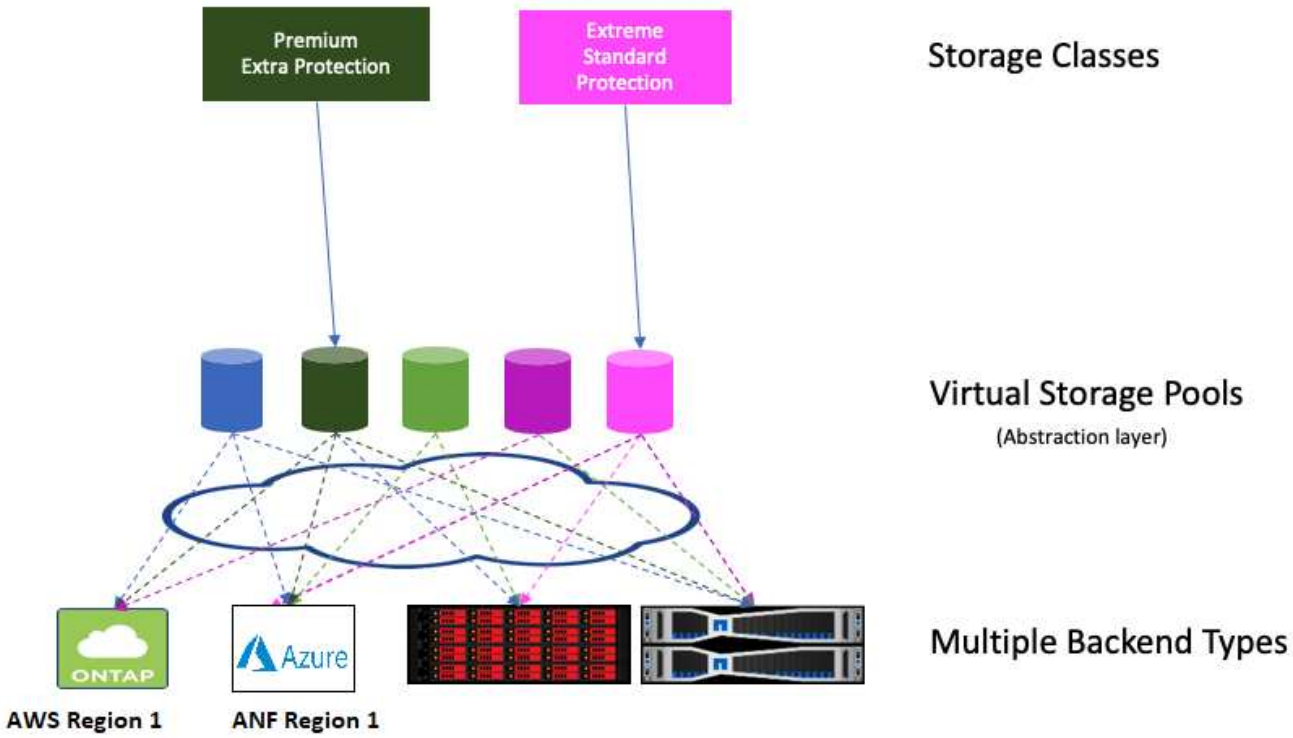
Com Trident, você pode usar VolumeSnapshots para criar novos PVs a partir deles. A criação de PVs a partir desses snapshots é realizada usando a tecnologia FlexClone para backends ONTAP compatíveis. Ao criar um PV a partir de um snapshot, o volume de suporte é um FlexClone do volume pai do snapshot. O `solidfire-san` driver usa clones de volume do software Element para criar PVs a partir de snapshots. Aqui, ele cria um clone a partir do snapshot do Element.

Pools virtuais

Os pools virtuais fornecem uma camada de abstração entre os backends de armazenamento do Trident e o Kubernetes `StorageClasses`. Eles permitem que um administrador defina aspectos, como localização, desempenho e proteção para cada backend de uma maneira comum e independente do backend, sem precisar especificar qual backend físico, pool de backends ou tipo de backend usar para atender aos critérios desejados. `StorageClass`

Saiba mais sobre pools virtuais

O administrador de storage pode definir pools virtuais em qualquer um dos backends do Trident em um arquivo de definição JSON ou YAML.



Qualquer aspecto especificado fora da lista de pools virtuais é global para o backend e será aplicado a todos os pools virtuais, enquanto cada pool virtual pode especificar um ou mais aspectos individualmente (substituindo qualquer aspecto global do backend).

- Ao definir pools virtuais, não tente reorganizar a ordem dos pools virtuais existentes em uma definição de backend.
- Recomendamos não modificar os atributos de um pool virtual existente. Você deve definir um novo pool virtual para efetuar as alterações.

A maioria dos aspectos é especificada em termos específicos do backend. É crucial notar que os valores dos aspectos não são expostos fora do driver do backend e não estão disponíveis para correspondência em `StorageClasses`. Em vez disso, o administrador define um ou mais rótulos para cada pool virtual. Cada rótulo é um par chave:valor, e os rótulos podem ser comuns entre backends distintos. Assim como os aspectos, os rótulos podem ser especificados por pool ou globalmente para o backend. Ao contrário dos aspectos, que possuem nomes e valores predefinidos, o administrador tem total liberdade para definir as chaves e os valores dos rótulos conforme necessário. Para maior conveniência, administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

Os rótulos do pool virtual podem ser definidos usando estes caracteres:

- letras maiúsculas `A-Z`
- letras minúsculas `a-z`
- números `0-9`
- sublinhados `_`
- hífens `-`

A `StorageClass` identifica qual pool virtual usar, referenciando os rótulos dentro de um parâmetro seletor. Seletores de pool virtual suportam os seguintes operadores:

Operador	Exemplo	O valor do rótulo de um pool deve:
`=`	performance=premium	Corresponder
`!=`	performance!=extremo	Não corresponde
`in`	localização em (leste, oeste)	Estar no conjunto de valores
`notin`	performance notin (prata, bronze)	Não estar no conjunto de valores
`<key>`	proteção	Existir com qualquer valor
`!<key>`	!protection	Não existe

Grupos de acesso a volume

Saiba mais sobre como Trident utiliza "grupos de acesso a volume".

Ignore esta seção se estiver usando CHAP, que é recomendado para simplificar o gerenciamento e evitar o limite de escalabilidade descrito abaixo. Além disso, se você estiver usando Trident no modo CSI, pode ignorar esta seção. Trident usa CHAP quando instalado como um provisionador CSI aprimorado.

Saiba mais sobre grupos de acesso de volume

Trident pode usar grupos de acesso a volumes para controlar o acesso aos volumes que provisiona. Se CHAP estiver desativado, ele espera encontrar um grupo de acesso chamado `trident` a menos que você especifique um ou mais IDs de grupo de acesso na configuração.

Embora Trident associe novos volumes aos grupos de acesso configurados, ele não cria nem gerencia os próprios grupos de acesso. Os grupos de acesso devem existir antes que o backend de storage seja adicionado ao Trident e precisam conter os IQNs iSCSI de todos os nós do cluster Kubernetes que possam potencialmente montar os volumes provisionados por esse backend. Na maioria das instalações, isso inclui todos os nós de trabalho do cluster.

Para clusters Kubernetes com mais de 64 nós, você deve usar vários grupos de acesso. Cada grupo de acesso pode conter até 64 IQNs, e cada volume pode pertencer a quatro grupos de acesso. Com o máximo de quatro grupos de acesso configurados, qualquer nó em um cluster de até 256 nós poderá acessar qualquer volume. Para os limites mais recentes de grupos de acesso a volumes, consulte "aqui".

Se você estiver modificando a configuração de uma que usa o grupo de acesso padrão `trident` para uma que usa outros também, inclua o ID do grupo de acesso `trident` na lista.

Início rápido para Trident

Você pode instalar Trident e começar a gerenciar recursos de storage em poucos passos. Antes de começar, revise "Requisitos do Trident".

> Para Docker, consulte "Trident for Docker".

1 Prepare o nó de trabalho

Todos os nós de trabalho no cluster Kubernetes devem ser capazes de montar os volumes que você provisionou para seus pods.

"Prepare o nó de trabalho"

2 Instalar Trident

Trident oferece vários métodos e modos de instalação otimizados para uma variedade de ambientes e organizações.

"Instale Trident"

3 Crie um backend

Um backend define a relação entre Trident e um sistema de storage. Ele informa Trident como se comunicar com esse sistema de storage e como Trident deve provisionar volumes a partir dele.

"Configurar um backend" para o seu sistema de storage

4 Criar um Kubernetes StorageClass

O objeto StorageClass do Kubernetes especifica Trident como provisionador e permite criar uma classe de armazenamento para provisionar volumes com atributos personalizáveis. Trident cria uma classe de armazenamento correspondente para objetos do Kubernetes que especificam o provisionador Trident.

"Crie uma storage class"

5 **Provisionar um volume**

Um *PersistentVolume* (PV) é um recurso de storage físico provisionado pelo administrador do cluster em um cluster Kubernetes. O *PersistentVolumeClaim* (PVC) é uma solicitação de acesso ao PersistentVolume no cluster.

Crie um PersistentVolume (PV) e um PersistentVolumeClaim (PVC) que utiliza o StorageClass do Kubernetes configurado para solicitar acesso ao PV. Em seguida, você pode montar o PV em um pod.

"Provisionar um volume"

Qual é o próximo passo?

Agora você pode adicionar backends adicionais, gerenciar classes de storage, gerenciar backends e realizar operações de volume.

Requisitos

Antes de instalar Trident, você deve revisar estes requisitos gerais do sistema. Backends específicos podem ter requisitos adicionais.

Informações essenciais sobre Trident

Você deve ler as seguintes informações críticas sobre Trident.

Informações críticas sobre Trident

- Kubernetes 1.34 agora é compatível com Trident. Atualize Trident antes de atualizar Kubernetes.
- Trident impõe rigorosamente o uso da configuração de multipath em ambientes SAN, com um valor recomendado de `find_multipaths: no` no arquivo multipath.conf.

  O uso de configuração sem multipath ou o uso de `find_multipaths: yes` ou `find_multipaths: smart` valor no arquivo multipath.conf resultará em falhas de montagem. Trident recomenda o uso de `find_multipaths: no` desde o lançamento 21.07.

Frontends suportados (orquestradores)

Trident suporta vários mecanismos de contêineres e orquestradores, incluindo os seguintes:

- Anthos On-Prem (VMware) e Anthos on bare metal 1.16

- Kubernetes 1.27 - 1.34
- OpenShift 4.12, 4.14 - 4.20 (se você planeja usar a preparação de nó iSCSI com OpenShift 4.19, a versão mínima do Trident suportada é 25.06.1.)

> Trident continua a oferecer suporte a versões mais antigas do OpenShift em alinhamento com o "Ciclo de vida do Red Hat Extended Update Support (EUS)", mesmo que elas dependam de versões do Kubernetes que não são mais oficialmente suportadas upstream. Ao instalar Trident nesses casos, você pode ignorar com segurança quaisquer mensagens de aviso sobre a versão do Kubernetes.

- Rancher Kubernetes Engine 2 (RKE2) v1.28.x - 1.34.x

> *Embora Trident seja compatível com as versões 1.27.x a 1.34.x do Rancher Kubernetes Engine 2 (RKE2), Trident foi qualificado atualmente apenas na versão 1.28.5+rke2r1 do RKE2.*

Trident também funciona com uma série de outras ofertas de Kubernetes totalmente gerenciadas e autogerenciadas, incluindo Google Kubernetes Engine (GKE), Amazon Elastic Kubernetes Services (EKS), Azure Kubernetes Service (AKS), Mirantis Kubernetes Engine (MKE) e VMWare Tanzu Portfolio.

Trident e ONTAP podem ser usados como provedores de storage para "KubeVirt".

> Antes de atualizar um cluster Kubernetes de 1.25 para 1.26 ou posterior que tenha Trident instalado, consulte "Atualize uma instalação Helm".

Backends suportados (storage)

Para usar Trident, você precisa de um ou mais dos seguintes backends suportados:

- Amazon FSx for NetApp ONTAP
- Azure NetApp Files
- Cloud Volumes ONTAP
- Google Cloud NetApp Volumes
- NetApp All SAN Array (ASA)
- FAS, AFF ou ASA r2 locais (iSCSI, NVMe/TCP e FC) executando versões do ONTAP sob suporte total ou limitado da NetApp. Consulte "Suporte à versão do software".
- NetApp HCI/Element software versão 11 ou superior

Suporte do Trident para KubeVirt e OpenShift Virtualization

Drivers de storage suportados:

Trident suporta os seguintes drivers ONTAP para KubeVirt e OpenShift Virtualization:

- ontap-nas
- ontap-nas-economy
- ontap-san (iSCSI, FCP, NVMe over TCP)
- ontap-san-economy (somente iSCSI)

Pontos a considerar:

- Atualize a classe de armazenamento para ter o `fsType` parâmetro (por exemplo: `fsType: "ext4"`) no ambiente de OpenShift Virtualization. Se necessário, defina o modo de volume como bloco explicitamente usando o `volumeMode=Block` parâmetro no `dataVolumeTemplates` para notificar o CDI para criar volumes de dados em bloco.
- *Modo de acesso RWX para drivers de armazenamento em bloco*: os drivers ontap-san (iSCSI, NVMe/TCP, FC) e ontap-san-economy (iSCSI) são suportados apenas com "volumeMode: Block" (dispositivo bruto). Para esses drivers, o `fstype`parâmetro não pode ser usado porque os volumes são fornecidos no modo de dispositivo bruto.
- Para fluxos de trabalho de migração em tempo real/s onde o modo de acesso RWX é exigido, estas combinações são suportadas:
  - NFS + `volumeMode=Filesystem`
  - iSCSI + `volumeMode=Block` (dispositivo bruto)
  - NVMe/TCP + `volumeMode=Block` (dispositivo bruto)
  - FC + `volumeMode=Block` (dispositivo bruto)

Requisitos de funcionalidade

A tabela abaixo resume os recursos disponíveis nesta versão do Trident e as versões do Kubernetes que ela suporta.

Recurso	Versão do Kubernetes	Gates de recursos necessários?
Trident	1.27 - 1.34	Não
Instantâneos de volume	1.27 - 1.34	Não
PVC de instantâneos de volume	1.27 - 1.34	Não
Redimensionamento de PV iSCSI	1.27 - 1.34	Não
CHAP bidirecional do ONTAP	1.27 - 1.34	Não
Políticas de exportação dinâmicas	1.27 - 1.34	Não
Trident Operator	1.27 - 1.34	Não
Topologia CSI	1.27 - 1.34	Não

Sistemas operacionais de host testados

Embora Trident não ofereça suporte oficial a sistemas operacionais específicos, os seguintes são conhecidos por funcionar:

- Versões do Red Hat Enterprise Linux CoreOS (RHCOS) suportadas pela OpenShift Container Platform em AMD64 e ARM64

- Red Hat Enterprise Linux (RHEL) 8 ou posterior em AMD64 e ARM64

  NVMe/TCP requer RHEL 9 ou posterior.

- Ubuntu 22.04 LTS ou posterior em AMD64 e ARM64
- Windows Server 2022
- SUSE Linux Enterprise Server (SLES) 15 ou posterior

Por padrão, Trident é executado em um contêiner e, portanto, funcionará em qualquer worker Linux. No entanto, esses workers precisam ser capazes de montar os volumes que o Trident fornece usando o cliente NFS padrão ou o iniciador iSCSI, dependendo dos backends que você estiver usando.

O `tridentctl` utilitário também funciona em qualquer uma dessas distribuições de Linux.

Configuração do host

Todos os nós de trabalho no cluster Kubernetes devem ser capazes de montar os volumes que você provisionou para seus pods. Para preparar os nós de trabalho, você deve instalar as ferramentas NFS, iSCSI ou NVMe, dependendo da sua seleção de driver.

"Prepare o nó de trabalho"

Configuração do sistema de storage

Trident pode exigir alterações em um sistema de storage antes que uma configuração de backend possa usá-lo.

"Configurar backends"

Portas Trident

Trident requer acesso a portas específicas para comunicação.

"Portas Trident"

Imagens de contêiner e versões correspondentes do Kubernetes

Para instalações isoladas da internet (air-gapped), a lista a seguir serve como referência para as imagens de contêiner necessárias para instalar Trident. Use o comando `tridentctl images` para verificar a lista de imagens de contêiner necessárias.

Imagens de contêiner necessárias para Trident 25.10

Versões do Kubernetes	Imagem do contêiner
v1.27.0, v1.28.0, v1.29.0, v1.30.0, v1.31.0, v1.32.0, v1.33.0, v1.34.0	• docker.io/netapp/trident:25.10.0 • docker.io/netapp/trident-autosupport:25.10 • registry.k8s.io/sig-storage/csi-provisioner:v5.3.0 • registry.k8s.io/sig-storage/csi-attacher:v4.10.0 • registry.k8s.io/sig-storage/csi-resizer:v1.14.0 • registry.k8s.io/sig-storage/csi-snapshotter:v8.3.0 • registry.k8s.io/sig-storage/csi-node-driver-registrar:v2.15.0 • docker.io/netapp/trident-operator:25.10.0 (opcional)

Instale Trident

Instale usando o operador Trident

Instalar usando o tridentctl

Instale usando o operador certificado OpenShift

Usar o Trident

Prepare o nó de trabalho

Todos os nós de trabalho no cluster Kubernetes devem ser capazes de montar os volumes que você provisionou para seus pods. Para preparar os nós de trabalho, você deve instalar as ferramentas NFS, iSCSI, NVMe/TCP ou FC, dependendo da seleção do driver.

Selecionando as ferramentas certas

Se você estiver usando uma combinação de drivers, deverá instalar todas as ferramentas necessárias para seus drivers. Versões recentes do Red Hat Enterprise Linux CoreOS (RHCOS) têm as ferramentas instaladas por padrão.

Ferramentas NFS

"Instale as ferramentas NFS" se você estiver usando: `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, ou `azure-netapp-files`.

Ferramentas iSCSI

"Instale as ferramentas iSCSI" se você estiver usando: `ontap-san`, `ontap-san-economy`, `solidfire-san`.

Ferramentas NVMe

"Instale as ferramentas NVMe" se você estiver usando `ontap-san` para nonvolatile memory express (NVMe) sobre o protocolo TCP (NVMe/TCP).

> NetApp recomenda ONTAP 9.12 ou posterior para NVMe/TCP.

Ferramentas SCSI sobre FC

Consulte "Formas de configurar hosts SAN FC e FC-NVMe" para obter mais informações sobre como configurar seus hosts SAN FC e FC-NVMe.

"Instale as ferramentas FC" se você estiver usando `ontap-san` com sanType `fcp` (SCSI sobre FC).

Pontos a considerar: * SCSI sobre FC é suportado em OpenShift e KubeVirt ambientes. * SCSI sobre FC não é suportado no Docker. * A autorrecuperação iSCSI não se aplica ao SCSI sobre FC.

Ferramentas SMB

"Prepare-se para provisionar volumes SMB" se você estiver usando: `ontap-nas` para provisionar volumes SMB.

Descoberta de serviço de nó

Trident tenta detectar automaticamente se o nó pode executar serviços iSCSI ou NFS.

> A descoberta de serviços de nó identifica os serviços descobertos, mas não garante que estejam configurados corretamente. Por outro lado, a ausência de um serviço descoberto não garante que a montagem do volume falhará.

Revisar eventos

Trident cria eventos para o nó a fim de identificar os serviços descobertos. Para revisar esses eventos, execute:

```
kubectl get event -A --field-selector involvedObject.name=<Kubernetes node name>
```

Revisar serviços descobertos

Trident identifica os serviços habilitados para cada nó no Trident node CR. Para visualizar os serviços descobertos, execute:

```
tridentctl get node -o wide -n <Trident namespace>
```

Volumes NFS

Instale as ferramentas NFS usando os comandos para o seu sistema operacional. Certifique-se de que o serviço NFS seja iniciado durante a inicialização.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Reinicie os nós de trabalho após instalar as ferramentas NFS para evitar falhas ao conectar volumes aos contêineres.

volumes iSCSI

Trident pode estabelecer automaticamente uma sessão iSCSI, analisar LUNs, descobrir dispositivos multipath, formatá-los e montá-los em um pod.

Capacidades de autorrecuperação do iSCSI

Para sistemas ONTAP, Trident executa a autorrecuperação iSCSI a cada cinco minutos para:

1. **Identifique** o estado desejado da sessão iSCSI e o estado atual da sessão iSCSI.
2. **Compare** o estado desejado com o estado atual para identificar os reparos necessários. Trident determina as prioridades de reparo e quando antecipar os reparos.
3. **Executar os reparos** necessários para retornar o estado atual da sessão iSCSI ao estado desejado da sessão iSCSI.

Os registros de atividades de autorrecuperação estão localizados no contêiner `trident-main` no respectivo pod do Daemonset. Para visualizar os registros, você deve ter definido `debug` como "true" durante a instalação do Trident.

As capacidades de autorrecuperação do Trident iSCSI podem ajudar a prevenir:

- Sessões iSCSI obsoletas ou com problemas de funcionamento que podem ocorrer após uma falha de conectividade de rede. No caso de uma sessão obsoleta, Trident aguarda sete minutos antes de encerrar a sessão para restabelecer a conexão com um portal.

  Por exemplo, se os segredos CHAP forem rotacionados no controlador de armazenamento e a rede perder a conectividade, os segredos CHAP antigos (desatualizados) podem persistir. A autorrecuperação pode reconhecer isso e restabelecer automaticamente a sessão para aplicar os segredos CHAP atualizados.

- Sessões iSCSI ausentes
- LUNs ausentes

Pontos a considerar antes de atualizar Trident

- Se apenas igroups por nó (introduzidos na 23.04+) estiverem em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para todos os dispositivos no barramento SCSI.
- Se apenas igroups com escopo de backend (obsoletos a partir da versão 23.04) estiverem em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para os IDs de LUN exatos no barramento SCSI.
- Se uma combinação de igroups por nó e igroups com escopo de backend estiver em uso, a autorrecuperação iSCSI iniciará novas varreduras SCSI para os IDs de LUN exatos no barramento SCSI.

Instale as ferramentas iSCSI

Instale as ferramentas iSCSI usando os comandos para o seu sistema operacional.

Antes de começar

- Cada nó no cluster Kubernetes deve ter um IQN único. **Este é um pré-requisito necessário**.
- Se estiver usando o RHCOS versão 4.5 ou posterior, ou outra distribuição Linux compatível com RHEL, com o `solidfire-san` driver e Element OS 12.5 ou anterior, certifique-se de que o algoritmo de autenticação CHAP esteja definido como MD5 em `/etc/iscsi/iscsid.conf`. Algoritmos CHAP seguros compatíveis com FIPS, SHA1, SHA-256 e SHA3-256 estão disponíveis com Element 12.7.

```
sudo sed -i 's/^\(node.session.auth.chap_algs\).*/\1 = MD5/'
/etc/iscsi/iscsid.conf
```

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com iSCSI PVs, especifique o `discard` mountOption em StorageClass para realizar exigência de espaço inline. Consulte "Documentação Red Hat".
- Certifique-se de ter atualizado para a versão mais recente do `multipath-tools`.

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils device-mapper-
multipath
```

2. Verifique se a versão do iscsi-initiator-utils é 6.2.0.874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Defina a digitalização como manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `iscsid` e `multipathd` estejam em execução:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão do open-iscsi é 2.0.874-5ubuntu2.10 ou posterior (para bionic) ou 2.0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Defina a digitalização como manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `open-iscsi` e `multipath-tools` estejam habilitados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

> Para o Ubuntu 18.04, você deve descobrir as portas de destino com `iscsiadm` antes de iniciar `open-iscsi` para que o daemon iSCSI seja iniciado. Você pode, alternativamente, modificar o serviço `iscsi` para iniciar `iscsid` automaticamente.

Configurar ou desativar autorrecuperação iSCSI

Você pode configurar as seguintes configurações de autorrecuperação do Trident iSCSI para corrigir sessões obsoletas:

- **Intervalo de autorrecuperação do iSCSI**: Determina a frequência com que a autorrecuperação do iSCSI é acionada (padrão: 5 minutos). Você pode configurá-lo para ser executado com mais frequência definindo um número menor ou com menos frequência definindo um número maior.

Definir o intervalo de autorrecuperação do iSCSI para 0 interrompe a autorrecuperação do iSCSI completamente. Não recomendamos desativar a autorrecuperação do iSCSI; ela só deve ser desativada em certos cenários, quando a autorrecuperação do iSCSI não estiver funcionando conforme o esperado ou para fins de depuração.

- **iSCSI Self-Healing Wait Time**: Determina a duração que a autorrecuperação do iSCSI aguarda antes de fazer logout de uma sessão com problemas e tentar fazer login novamente (padrão: 7 minutos). Você pode configurar para um número maior, de modo que as sessões identificadas como com problemas tenham que esperar mais tempo antes de serem desconectadas e, em seguida, uma tentativa de login seja feita, ou para um número menor para fazer logout e login mais cedo.

Helm

Para configurar ou alterar as configurações de autorrecuperação do iSCSI, passe os `iscsiSelfHealingInterval` e `iscsiSelfHealingWaitTime` parâmetros durante a instalação ou atualização do helm.

O exemplo a seguir define o intervalo de autorrecuperação do iSCSI para 3 minutos e o tempo de espera para autorrecuperação para 6 minutos:

```
helm install trident trident-operator-100.2506.0.tgz --set
iscsiSelfHealingInterval=3m0s --set iscsiSelfHealingWaitTime=6m0s -n
trident
```

tridentctl

Para configurar ou alterar as configurações de autorrecuperação do iSCSI, passe os `iscsi-self-healing-interval` e `iscsi-self-healing-wait-time` parâmetros durante a instalação ou atualização do tridentctl.

O exemplo a seguir define o intervalo de autorrecuperação do iSCSI para 3 minutos e o tempo de espera para autorrecuperação para 6 minutos:

```
tridentctl install --iscsi-self-healing-interval=3m0s --iscsi-self
-healing-wait-time=6m0s -n trident
```

Volumes NVMe/TCP

Instale as ferramentas NVMe usando os comandos para o seu sistema operacional.

- NVMe requer RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a sua versão do kernel, talvez seja necessário atualizar a versão do kernel do seu nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Verificar instalação

Após a instalação, verifique se cada nó no cluster Kubernetes possui um NQN exclusivo usando o comando:

```
cat /etc/nvme/hostnqn
```

Trident modifica o `ctrl_device_tmo` valor para garantir que o NVMe não abandone o caminho caso ocorra uma falha. Não altere essa configuração.

Volumes SCSI over FC

Agora você pode usar o protocolo Fibre Channel (FC) com Trident para provisionar e gerenciar recursos de storage no sistema ONTAP.

Pré-requisitos

Configurar as definições de rede e de nó necessárias para FC.

Configurações de rede

1. Obtenha o WWPN das interfaces de destino. Consulte "network interface show" para mais informações.
2. Obtenha o WWPN para as interfaces no iniciador (Host).

   Consulte os utilitários correspondentes do sistema operacional do host.

3. Configurar o zoneamento no switch FC usando os WWPNs do host e do destino.

   Consulte a documentação do respectivo fornecedor do switch para obter informações.

   Consulte a seguinte documentação do ONTAP para obter detalhes:

   - "Visão geral do zoneamento Fibre Channel e FCoE"
   - "Formas de configurar hosts SAN FC e FC-NVMe"

Instale as ferramentas FC

Instale as ferramentas FC usando os comandos para o seu sistema operacional.

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com FC PVs, especifique o `discard` mountOption na StorageClass para realizar exigência de espaço inline. Consulte "Documentação Red Hat".

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipathd` está em execução:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `/etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipath-tools` está ativado e em execução:

```
sudo systemctl status multipath-tools
```

Prepare-se para provisionar volumes SMB

Você pode provisionar volumes SMB usando `ontap-nas` drivers.

> Você deve configurar ambos os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters ONTAP locais. A falha ao configurar qualquer um desses protocolos fará com que a criação do volume SMB falhe.

> `autoExportPolicy` não é compatível com volumes SMB.

Antes de começar

Antes de poder provisionar volumes SMB, você deve ter o seguinte.

- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo Trident contendo suas credenciais do Active Directory. Para gerar o segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Passos

1. Para o ONTAP local, você pode opcionalmente criar um compartilhamento SMB ou o Trident pode criar um para você.

   > Os compartilhamentos SMB são necessários para Amazon FSx for ONTAP.

   Você pode criar os compartilhamentos administrativos SMB de duas maneiras: usando o "Microsoft Management Console" snap-in Shared Folders ou usando a ONTAP CLI. Para criar os compartilhamentos SMB usando a ONTAP CLI:

   a. Se necessário, crie a estrutura de caminho de diretórios para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação do compartilhamento. Se o caminho especificado não existir, o comando falha.

   b. Crie um compartilhamento SMB associado à SVM especificada:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

Consulte "Criar um compartilhamento SMB" para obter detalhes completos.

2. Ao criar o backend, você deve configurar o seguinte para especificar os volumes SMB. Para todas as opções de configuração do backend do FSx para ONTAP, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`nasType`	**Deve ser definido como `smb`.** Se for nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` ou `mixed` for SMB volumes
`unixPermissions`	Modo para novos volumes. **Deve ser deixado em branco para volumes SMB.**	""

Configurar e gerenciar backends

Configurar backends

Um backend define a relação entre Trident e um sistema de storage. Ele informa Trident como se comunicar com esse sistema de storage e como Trident deve provisionar volumes a partir dele.

Trident oferece automaticamente pools de armazenamento de backends que correspondem aos requisitos definidos por uma classe de armazenamento. Saiba como configurar o backend para o seu sistema de storage.

- "Configurar um backend do Azure NetApp Files"
- "Configurar um backend do Google Cloud NetApp Volumes"
- "Configurar um NetApp HCI ou SolidFire backend"
- "Configure um backend com os drivers NAS do ONTAP ou do Cloud Volumes ONTAP"

- "Configure um backend com drivers SAN do ONTAP ou Cloud Volumes ONTAP"
- "Use Trident com Amazon FSx for NetApp ONTAP"

Azure NetApp Files

Configurar um backend do Azure NetApp Files

Você pode configurar o Azure NetApp Files como o backend para Trident. Você pode anexar volumes NFS e SMB usando um backend do Azure NetApp Files. Trident também oferece suporte ao gerenciamento de credenciais usando identidades gerenciadas para clusters do Azure Kubernetes Services (AKS).

Detalhes do driver do Azure NetApp Files

Trident fornece os seguintes drivers de armazenamento Azure NetApp Files para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`azure-netapp-files`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Considerações

- O serviço Azure NetApp Files não suporta volumes menores que 50 GiB. Trident cria automaticamente volumes de 50 GiB se um volume menor for solicitado.
- Trident suporta volumes SMB montados em pods em execução apenas em nós Windows.

Identidades gerenciadas para AKS

Trident oferece suporte "identidades gerenciadas" para clusters do Azure Kubernetes Services. Para aproveitar o gerenciamento simplificado de credenciais oferecido pelas identidades gerenciadas, você deve ter:

- Um cluster Kubernetes implantado usando AKS
- Identidades gerenciadas configuradas no cluster kubernetes do AKS
- Trident instalado que inclui o `cloudProvider` para especificar `"Azure"`.

Operador Trident

Para instalar Trident usando o Trident operator, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` para `"Azure"`. Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
```

Helm

O exemplo a seguir instala conjuntos Trident `cloudProvider` no Azure usando a variável de ambiente `$CP`:

```
helm install trident trident-operator-100.2506.0.tgz --create
-namespace --namespace <trident-namespace> --set cloudProvider=$CP
```

<code>tridentctl</code>

O exemplo a seguir instala Trident e define o `cloudProvider` sinalizador para `Azure`:

```
tridentctl install --cloud-provider="Azure" -n trident
```

Identidade na nuvem para AKS

A identidade na nuvem permite que os pods do Kubernetes acessem recursos do Azure autenticando-se como uma workload identity em vez de fornecer credenciais explícitas do Azure.

Para aproveitar as vantagens da identidade na nuvem no Azure, você deve ter:

- Um cluster Kubernetes implantado usando AKS
- Identidade de carga de trabalho e oidc-issuer configurados no cluster Kubernetes do AKS
- Trident instalado que inclui o `cloudProvider` para especificar `"Azure"` e `cloudIdentity` especificando a identidade da carga de trabalho

Operador Trident

Para instalar Trident usando o Trident operator, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` para `"Azure"` e defina `cloudIdentity` para `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "Azure"
  cloudIdentity: 'azure.workload.identity/client-id: xxxxxxxx-xxxx-
xxxx-xxxx-xxxxxxxxxxx' # Edit
```

Helm

Defina os valores para as flags **cloud-provider (CP)** e **cloud-identity (CI)** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="'azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx'"
```

O exemplo a seguir instala Trident e define `cloudProvider` para Azure usando a variável de ambiente `$CP` e define o `cloudIdentity` usando a variável de ambiente `$CI`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$CI"
```

<code>tridentctl</code>

Defina os valores para os parâmetros **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="Azure"
export CI="azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-
xxxxxxxxxxx"
```

O exemplo a seguir instala Trident e define o `cloud-provider` sinalizador para `$CP`, e `cloud-identity` para `$CI`:

```
tridentctl install --cloud-provider=$CP --cloud-identity="$CI" -n
trident
```

Prepare-se para configurar um backend do Azure NetApp Files

Antes de configurar seu backend do Azure NetApp Files, você precisa garantir que os seguintes requisitos sejam atendidos.

Pré-requisitos para volumes NFS e SMB

Se você estiver usando o Azure NetApp Files pela primeira vez ou em um novo local, será necessária alguma configuração inicial para configurar o Azure NetApp Files e criar um volume NFS. Consulte "Azure: configure o Azure NetApp Files e crie um volume NFS".

Para configurar e usar um "Azure NetApp Files" backend, você precisa do seguinte:

> - `subscriptionID`, `tenantID`, `clientID`, `location` e `clientSecret` são opcionais ao usar identidades gerenciadas em um cluster AKS.
> - `tenantID`, `clientID`, e `clientSecret` são opcionais ao usar uma identidade de nuvem em um cluster AKS.

- Um pool de capacidade. Consulte "Microsoft: criar um pool de capacidade para Azure NetApp Files".
- Uma sub-rede delegada ao Azure NetApp Files. Consulte "Microsoft: delegar uma sub-rede ao Azure NetApp Files".
- `subscriptionID` de uma assinatura do Azure com o Azure NetApp Files ativado.
- `tenantID`, `clientID`, e `clientSecret` de um "Registro de aplicativo" no Azure Active Directory com permissões suficientes para o serviço NetApp Files do Azure. O registro do aplicativo deve usar um dos seguintes:
  - O papel de Owner ou Contributor "predefinido pelo Azure".
  - Uma "Função de Contributor personalizada" no nível da assinatura (`assignableScopes` com as seguintes permissões, que são limitadas apenas ao que Trident exige. Após criar a função personalizada, "atribua a função usando o portal do Azure".

Função de colaborador personalizado

```
{
  "id": "/subscriptions/<subscription-
id>/providers/Microsoft.Authorization/roleDefinitions/<role-
definition-id>",
  "properties": {
    "roleName": "custom-role-with-limited-perms",
    "description": "custom role providing limited permissions",
    "assignableScopes": [
      "/subscriptions/<subscription-id>"
    ],
    "permissions": [
      {
        "actions": [
          "Microsoft.NetApp/netAppAccounts/capacityPools/read",
          "Microsoft.NetApp/netAppAccounts/capacityPools/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
read",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
write",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/snapshots/
delete",

"Microsoft.NetApp/netAppAccounts/capacityPools/volumes/MountTarge
ts/read",
          "Microsoft.Network/virtualNetworks/read",
          "Microsoft.Network/virtualNetworks/subnets/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/read",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/write",

"Microsoft.Features/featureProviders/subscriptionFeatureRegistrat
ions/delete",
```

```
            "Microsoft.Features/features/read",
            "Microsoft.Features/operations/read",
            "Microsoft.Features/providers/features/read",

"Microsoft.Features/providers/features/register/action",

"Microsoft.Features/providers/features/unregister/action",

"Microsoft.Features/subscriptionFeatureRegistrations/read"
          ],
          "notActions": [],
          "dataActions": [],
          "notDataActions": []
        }
      ]
    }
}
```

- O Azure `location` que contém pelo menos um "sub-rede delegada". A partir do Trident 22.01, o `location` parâmetro é um campo obrigatório no nível superior do arquivo de configuração do backend. Os valores de localização especificados em pools virtuais são ignorados.
- Para usar `Cloud Identity`, obtenha o `client ID` de um "identidade gerenciada atribuída pelo usuário" e especifique esse ID em `azure.workload.identity/client-id: xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxx`.

Requisitos adicionais para volumes SMB

Para criar um volume SMB, você deve ter:

- Active Directory configurado e conectado ao Azure NetApp Files. Consulte "Microsoft: criar e gerenciar conexões do Active Directory para Azure NetApp Files".
- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo do Trident contendo suas credenciais do Active Directory para que o Azure NetApp Files possa autenticar no Active Directory. Para gerar o segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Opções e exemplos de configuração do backend do Azure NetApp Files

Saiba mais sobre as opções de configuração de back-end NFS e SMB para o Azure NetApp Files e veja exemplos de configuração.

Opções de configuração do backend

Trident usa sua configuração de back-end (sub-rede, rede virtual, nível de serviço e localização) para criar volumes do Azure NetApp Files em pools de capacidade disponíveis na localização solicitada e que correspondam ao nível de serviço e à sub-rede solicitados.

Os backends do Azure NetApp Files fornecem essas opções de configuração.

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	"azure-netapp-files"
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + caracteres aleatórios
`subscriptionID`	O ID da assinatura da sua assinatura do Azure opcional quando as identidades gerenciadas estão habilitadas em um cluster do AKS.	
`tenantID`	O ID do locatário de um registro de aplicativo opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`clientID`	O ID do cliente de um registro de aplicativo é opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`clientSecret`	O segredo do cliente de um registro de aplicativo. Opcional quando identidades gerenciadas ou identidade na nuvem são usadas em um cluster AKS.	
`serviceLevel`	Um de `Standard`, `Premium` ou `Ultra`	"" (aleatório)
`location`	Nome da localização do Azure onde os novos volumes serão criados Opcional quando as identidades gerenciadas estão habilitadas em um cluster AKS.	
`resourceGroups`	Lista de grupos de recursos para filtrar recursos descobertos	"[]" (sem filtro)

Parâmetro	Descrição	Padrão
`netappAccounts`	Lista de contas NetApp para filtrar recursos descobertos	"[]" (sem filtro)
`capacityPools`	Lista de pools de capacidade para filtrar recursos descobertos	"[]" (sem filtro, aleatório)
`virtualNetwork`	Nome de uma rede virtual com uma sub-rede delegada	""
`subnet`	Nome de uma sub-rede delegada a `Microsoft.Netapp/volumes`	""
`networkFeatures`	Conjunto de recursos de VNet para um volume, pode ser `Basic` ou `Standard`. Network Features não está disponível em todas as regiões e pode precisar ser habilitado em uma assinatura. Especificar `networkFeatures` quando a funcionalidade não está habilitada faz com que o provisionamento do volume falhe.	""
`nfsMountOptions`	Controle preciso das opções de montagem NFS. Ignorado para volumes SMB. Para montar volumes usando a versão de NFS 4.1, inclua `nfsvers=4` na lista de opções de montagem separadas por vírgula para escolher NFS v4.1. As opções de montagem definidas em uma definição de classe de armazenamento substituem as opções de montagem definidas na configuração do backend.	"nfsvers=3"
`limitVolumeSize`	Falhar no provisionamento se o tamanho do volume solicitado for superior a este valor	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, `\{"api": false, "method": true, "discovery": true}`. Não use isso a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nasType`	Configure a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null utiliza volumes NFS por padrão.	`nfs`

Parâmetro	Descrição	Padrão
`supportedTopologies`	Representa uma lista de regiões e zonas suportadas por este backend. Para mais informações, consulte "Usar a topologia CSI".	
`qosType`	Representa o tipo de QoS: automático ou manual.	Automático
`maxThroughput`	Define a taxa de transferência máxima permitida em MiB/s. Compatível apenas com pools de capacidade QoS manuais.	`4 MiB/sec`

Para mais informações sobre recursos de rede, consulte "Configurar recursos de rede para um volume do Azure NetApp Files".

Permissões e recursos necessários

Se você receber um erro "Nenhum pool de capacidade encontrado" ao criar um PVC, é provável que o registro do seu aplicativo não tenha as permissões e recursos necessários (sub-rede, rede virtual, pool de capacidade) associados. Se o modo de depuração estiver habilitado, Trident registrará os recursos do Azure descobertos quando o backend for criado. Verifique se uma função apropriada está sendo usada.

Os valores para `resourceGroups`, `netappAccounts`, `capacityPools`, `virtualNetwork` e `subnet` podem ser especificados usando nomes curtos ou totalmente qualificados. Nomes totalmente qualificados são recomendados na maioria das situações, pois nomes curtos podem corresponder a vários recursos com o mesmo nome.

Se a vNet estiver localizada em um grupo de recursos diferente da conta de armazenamento Azure NetApp Files (ANF), especifique o grupo de recursos para a rede virtual ao configurar a lista resourceGroups para o backend.

Os `resourceGroups`, `netappAccounts` e `capacityPools` valores são filtros que restringem o conjunto de recursos descobertos àqueles disponíveis para este backend de armazenamento e podem ser especificados em qualquer combinação. Os nomes totalmente qualificados seguem este formato:

Tipo	Formatar
Grupo de recursos	<resource group>
Conta do NetApp	<resource group>/<netapp account>
Pool de capacidade	<resource group>/<netapp account>/<capacity pool>
Rede virtual	<resource group>/<virtual network>
Sub-rede	<resource group>/<virtual network>/<subnet>

Provisionamento de volume

Você pode controlar o provisionamento de volumes padrão especificando as seguintes opções em uma seção especial do arquivo de configuração. Consulte Exemplos de configurações para obter detalhes.

Parâmetro	Descrição	Padrão
`exportRule`	Regras de exportação para novos volumes. `exportRule` deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4 ou sub-redes IPv4 na notação CIDR. Ignorada para volumes SMB.	"0.0.0.0/0"
`snapshotDir`	Controla a visibilidade do diretório .snapshot	"true" para NFSv4 "false" para NFSv3
`size`	O tamanho padrão de novos volumes	"100G"
`unixPermissions`	As permissões Unix de novos volumes (4 dígitos octais). Ignorado para volumes SMB.	"" (recurso em pré-visualização, requer inclusão na lista de permissões na assinatura)

Exemplos de configurações

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros com os valores padrão. Esta é a maneira mais fácil de definir um backend.

Configuração mínima

Esta é a configuração mínima absoluta de backend. Com esta configuração, Trident descobre todas as suas NetApp contas, pools de capacidade e sub-redes delegadas ao Azure NetApp Files no local configurado e coloca novos volumes em um desses pools e sub-redes aleatoriamente. Como `nasType` é omitido, o `nfs` padrão se aplica e o backend irá provisionar volumes NFS.

Essa configuração é ideal quando você está começando a usar o Azure NetApp Files e testando as funcionalidades, mas, na prática, você vai querer fornecer um escopo adicional para os volumes que você provisionar.

```
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
  tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
  clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
  clientSecret: SECRET
  location: eastus
```

Identidades gerenciadas para AKS

Esta configuração de backend omite `subscriptionID`, `tenantID`, `clientID` e `clientSecret`, que são opcionais ao usar identidades gerenciadas.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - resource-group-1/netapp-account-1/ultra-pool
  resourceGroups:
    - resource-group-1
  netappAccounts:
    -  resource-group-1/netapp-account-1
  virtualNetwork: resource-group-1/eastus-prod-vnet
  subnet: resource-group-1/eastus-prod-vnet/eastus-anf-subnet
```

Identidade na nuvem para AKS

Esta configuração de backend omite `tenantID`, `clientID` e `clientSecret`, que são opcionais ao usar uma identidade na nuvem.

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-anf-1
  namespace: trident
spec:
  version: 1
  storageDriverName: azure-netapp-files
  capacityPools:
    - ultra-pool
  resourceGroups:
    - aks-ami-eastus-rg
  netappAccounts:
    - smb-na
  virtualNetwork: eastus-prod-vnet
  subnet: eastus-anf-subnet
  location: eastus
  subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
```

Configuração específica de nível de serviço com filtros de pool de capacidade

Essa configuração de backend coloca volumes na localização do Azure `eastus` em um `Ultra` pool de capacidade. Trident descobre automaticamente todas as sub-redes delegadas ao Azure NetApp Files nesse local e coloca um novo volume em uma delas aleatoriamente.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
```

Exemplo de backend com pools de capacidade QoS manuais

Essa configuração de backend coloca volumes na localização do Azure `eastus` com pools de capacidade de QoS manuais.

```
---
version: 1
storageDriverName: azure-netapp-files
backendName: anf1
location: eastus
labels:
  clusterName: test-cluster-1
  cloud: anf
  nasType: nfs
defaults:
  qosType: Manual
storage:
  - serviceLevel: Ultra
    labels:
      performance: gold
    defaults:
      maxThroughput: 10
  - serviceLevel: Premium
    labels:
      performance: silver
    defaults:
      maxThroughput: 5
  - serviceLevel: Standard
    labels:
      performance: bronze
    defaults:
      maxThroughput: 3
```

Configuração avançada

Essa configuração de backend reduz ainda mais o escopo do posicionamento de volumes para uma única sub-rede e também modifica alguns padrões de provisionamento de volumes.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
virtualNetwork: application-group-1/eastus-prod-vnet
subnet: application-group-1/eastus-prod-vnet/my-subnet
networkFeatures: Standard
nfsMountOptions: vers=3,proto=tcp,timeo=600
limitVolumeSize: 500Gi
defaults:
  exportRule: 10.0.0.0/24,10.0.1.0/24,10.0.2.100
  snapshotDir: "true"
  size: 200Gi
  unixPermissions: "0777"
```

Configuração de pool virtual

Esta configuração de backend define vários pools de armazenamento em um único arquivo. Isso é útil quando você tem vários pools de capacidade que suportam diferentes níveis de serviço e deseja criar classes de armazenamento no Kubernetes que os representem. Rótulos de pool virtual foram usados para diferenciar os pools com base em `performance`.

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
resourceGroups:
  - application-group-1
networkFeatures: Basic
nfsMountOptions: vers=3,proto=tcp,timeo=600
labels:
  cloud: azure
storage:
  - labels:
      performance: gold
    serviceLevel: Ultra
    capacityPools:
      - application-group-1/netapp-account-1/ultra-1
      - application-group-1/netapp-account-1/ultra-2
    networkFeatures: Standard
  - labels:
      performance: silver
    serviceLevel: Premium
    capacityPools:
      - application-group-1/netapp-account-1/premium-1
  - labels:
      performance: bronze
    serviceLevel: Standard
    capacityPools:
      - application-group-1/netapp-account-1/standard-1
      - application-group-1/netapp-account-1/standard-2
```

Configuração de topologias suportadas

Trident facilita o provisionamento de volumes para cargas de trabalho com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de backend é usado para fornecer uma lista de regiões e zonas por backend. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria volumes na região e zona mencionadas. Para mais informações, consulte "Usar a topologia CSI".

```
---
version: 1
storageDriverName: azure-netapp-files
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: eastus
serviceLevel: Ultra
capacityPools:
  - application-group-1/account-1/ultra-1
  - application-group-1/account-1/ultra-2
supportedTopologies:
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-1
  - topology.kubernetes.io/region: eastus
    topology.kubernetes.io/zone: eastus-2
```

Definições de classe de armazenamento

As seguintes `StorageClass`definições referem-se aos pools de armazenamento acima.

Exemplo de definições usando `parameter.selector` field

Usando `parameter.selector` você pode especificar para cada `StorageClass` o pool virtual que é usado para hospedar um volume. O volume terá os aspectos definidos no pool escolhido.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gold
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
allowVolumeExpansion: true

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: bronze
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze
allowVolumeExpansion: true
```

Exemplos de definições para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do Active Directory.

Configuração básica no namespace padrão

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Usando segredos diferentes por namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Utilizando segredos diferentes em cada volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: anf-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "azure-netapp-files"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`Filtros para pools que suportam volumes SMB. `nasType: nfs` ou `nasType: null`filtros para pools NFS.

Criar o backend

Após criar o arquivo de configuração de backend, execute o seguinte comando:

```
tridentctl create backend -f <backend-file>
```

Se a criação do backend falhar, há algo errado com a configuração do backend. Você pode visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Google Cloud NetApp Volumes

Configurar um backend do Google Cloud NetApp Volumes

Agora você pode configurar o Google Cloud NetApp Volumes como backend para Trident. Você pode anexar volumes NFS e SMB usando um backend do Google Cloud NetApp Volumes.

Detalhes do driver do Google Cloud NetApp Volumes

Trident fornece o `google-cloud-netapp-volumes` driver para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`google-cloud-netapp-volumes`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	`nfs`, `smb`

Identidade na nuvem para GKE

A identidade na nuvem permite que os pods do Kubernetes acessem recursos do Google Cloud autenticando-se como uma workload identity, em vez de fornecer credenciais explícitas do Google Cloud.

Para aproveitar a identidade na nuvem do Google Cloud, você precisa ter:

- Um cluster Kubernetes implantado usando GKE.
- Identidade de carga de trabalho configurada no cluster GKE e GKE MetaData Server configurado nos pools de nós.

- Uma conta de serviço do GCP com a função Google Cloud NetApp Volumes Admin (roles/netapp.admin) ou uma função personalizada.
- Trident instalado que inclui o cloudProvider para especificar "GCP" e cloudIdentity especificando a nova conta de serviço do GCP. Um exemplo é fornecido abaixo.

Operador Trident

Para instalar Trident usando o Trident operator, edite `tridentorchestrator_cr.yaml` para definir `cloudProvider` para `"GCP"` e defina `cloudIdentity` para `iam.gke.io/gcp-service-account: cloudvolumes-admin-sa@mygcpproject.iam.gserviceaccount.com`.

Por exemplo:

```
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident
  imagePullPolicy: IfNotPresent
  cloudProvider: "GCP"
  cloudIdentity: 'iam.gke.io/gcp-service-account: cloudvolumes-
admin-sa@mygcpproject.iam.gserviceaccount.com'
```

Helm

Defina os valores para as flags **cloud-provider (CP)** e **cloud-identity (CI)** usando as seguintes variáveis de ambiente:

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

O exemplo a seguir instala Trident e define `cloudProvider` para GCP usando a variável de ambiente `$CP` e define o `cloudIdentity` usando a variável de ambiente `$ANNOTATION`:

```
helm install trident trident-operator-100.6.0.tgz --set
cloudProvider=$CP --set cloudIdentity="$ANNOTATION"
```

<code>tridentctl</code>

Defina os valores para os parâmetros **provedor de nuvem** e **identidade de nuvem** usando as seguintes variáveis de ambiente:

```
export CP="GCP"
export ANNOTATION="'iam.gke.io/gcp-service-account: cloudvolumes-admin-
sa@mygcpproject.iam.gserviceaccount.com'"
```

O exemplo a seguir instala Trident e define o `cloud-provider` sinalizador para `$CP`, e `cloud-identity` para `$ANNOTATION`:

```
tridentctl install --cloud-provider=$CP --cloud
-identity="$ANNOTATION" -n trident
```

Prepare-se para configurar um backend do Google Cloud NetApp Volumes

Antes de configurar seu backend do Google Cloud NetApp Volumes, você precisa garantir que os seguintes requisitos sejam atendidos.

Pré-requisitos para volumes NFS

Se você estiver usando o Google Cloud NetApp Volumes pela primeira vez ou em um novo local, é necessário realizar uma configuração inicial para configurar o Google Cloud NetApp Volumes e criar um volume NFS. Consulte "Antes de começar".

Certifique-se de ter o seguinte antes de configurar o backend do Google Cloud NetApp Volumes:

- Uma conta do Google Cloud configurada com o serviço Google Cloud NetApp Volumes. Consulte "Google Cloud NetApp Volumes".
- Número do projeto da sua conta do Google Cloud. Consulte "Identificação de projetos".
- Uma conta de serviço do Google Cloud com a função de NetApp Volumes Admin (`roles/netapp.admin`. Consulte "Funções e permissões de Identity and Access Management".
- Arquivo de chave API para sua conta GCNV. Consulte "Criar uma chave de conta de serviço"
- Um pool de storage. Consulte "Visão geral dos storage pools".

Para obter mais informações sobre como configurar o acesso ao Google Cloud NetApp Volumes, consulte "Configure o acesso ao Google Cloud NetApp Volumes".

Opções e exemplos de configuração do backend do Google Cloud NetApp Volumes

Saiba mais sobre as opções de configuração de back-end para Google Cloud NetApp Volumes e veja exemplos de configuração.

Opções de configuração do backend

Cada backend provisiona volumes em uma única região do Google Cloud. Para criar volumes em outras regiões, você pode definir backends adicionais.

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	O valor de `storageDriverName` deve ser especificado como "google-cloud-netapp-volumes".
`backendName`	(Opcional) Nome personalizado do storage backend	Nome do driver + "_" + parte da chave da API

Parâmetro	Descrição	Padrão
`storagePools`	Parâmetro opcional usado para especificar pools de storage para criação de volumes.	
`projectNumber`	Número do projeto da conta do Google Cloud. O valor é encontrado na página inicial do portal do Google Cloud.	
`location`	O local do Google Cloud onde Trident cria volumes GCNV. Ao criar clusters Kubernetes entre regiões, volumes criados em um `location` podem ser usados em cargas de trabalho agendadas em nós em várias regiões do Google Cloud. O tráfego entre regiões gera um custo adicional.	
`apiKey`	Chave de API para a conta de serviço do Google Cloud com a `netapp.admin` função. Inclui o conteúdo formatado em JSON do arquivo de chave privada de uma conta de serviço do Google Cloud (copiado integralmente para o arquivo de configuração do backend). O `apiKey` deve incluir pares de chave-valor para as seguintes chaves: `type`, `project_id`, `client_email`, `client_id`, `auth_uri`, `token_uri`, `auth_provider_x509_cert_url` e `client_x509_cert_url`.	
`nfsMountOptions`	Controle preciso das opções de montagem NFS.	"nfsvers=3"
`limitVolumeSize`	Falhe no provisionamento se o tamanho do volume solicitado for superior a esse valor.	"" (não aplicado por padrão)
`serviceLevel`	O nível de serviço de um pool de storage e seus volumes. Os valores são `flex`, `standard`, `premium`, ou `extreme`.	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`network`	Rede Google Cloud usada para volumes GCNV.	
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, `{"api":false, "method":true}`. Não use isso a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nasType`	Configure a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null utiliza volumes NFS por padrão.	`nfs`

Parâmetro	Descrição	Padrão
`supportedTopologies`	Representa uma lista de regiões e zonas suportadas por este backend. Para mais informações, consulte "Usar a topologia CSI". Por exemplo: `supportedTopologies: - topology.kubernetes.io/region: asia-east1 topology.kubernetes.io/zone: asia-east1-a`	

Opções de provisionamento de volume

Você pode controlar o provisionamento de volume padrão na seção `defaults` do arquivo de configuração.

Parâmetro	Descrição	Padrão
`exportRule`	As regras de exportação para novos volumes. Deve ser uma lista separada por vírgulas de qualquer combinação de endereços IPv4.	"0.0.0.0/0"
`snapshotDir`	Acesso ao `.snapshot` diretório	"true" para NFSv4 "false" para NFSv3
`snapshotReserve`	Percentual do volume reservado para snapshots	"" (aceitar padrão de 0)
`unixPermissions`	As permissões unix de novos volumes (4 dígitos octais).	""

Exemplos de configurações

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros com os valores padrão. Esta é a maneira mais fácil de definir um backend.

Configuração mínima

Esta é a configuração mínima absoluta de backend. Com esta configuração, Trident descobre todos os seus pools de storage delegados ao Google Cloud NetApp Volumes no local configurado e coloca novos volumes em um desses pools aleatoriamente. Como `nasType` foi omitido, o `nfs` padrão se aplica e o backend irá provisionar volumes NFS.

Essa configuração é ideal quando você está começando a usar o Google Cloud NetApp Volumes e testando as funcionalidades, mas, na prática, provavelmente será necessário fornecer um escopo adicional para os volumes que você provisionar.

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração para volumes SMB

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv1
  namespace: trident
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123456789"
  location: asia-east1
  serviceLevel: flex
  nasType: smb
  apiKey:
    type: service_account
    project_id: cloud-native-data
    client_email: trident-sample@cloud-native-
data.iam.gserviceaccount.com
    client_id: "123456789737813416734"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/trident-
sample%40cloud-native-data.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração com filtro StoragePools

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  serviceLevel: premium
  storagePools:
    - premium-pool1-europe-west6
    - premium-pool2-europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
```

Configuração de pool virtual

Esta configuração de backend define vários pools virtuais em um único arquivo. Os pools virtuais são definidos na `storage` seção. Eles são úteis quando você tem vários pools de storage suportando diferentes níveis de serviço e deseja criar classes de storage no Kubernetes que os representem. Os rótulos dos pools virtuais são usados para diferenciá-los. Por exemplo, no exemplo abaixo `performance` label e `serviceLevel` type são usados para diferenciar os pools virtuais.

Você também pode definir alguns valores padrão que serão aplicáveis a todos os pools virtuais e sobrescrever os valores padrão para pools virtuais individuais. No exemplo a seguir, `snapshotReserve` e `exportRule` servem como padrões para todos os pools virtuais.

Para obter mais informações, consulte "Pools virtuais".

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-gcnv-secret
type: Opaque
stringData:
  private_key_id: f2cb6ed6d7cc10c453f7d3406fc700c5df0ab9ec
  private_key: |
    -----BEGIN PRIVATE KEY-----
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    znHczZsrrtHisIsAbOguSaPIKeyAZNchRAGzlzZE4jK3bl/qp8B4Kws8zX5ojY9m
    XsYg6gyxy4zq7OlwWgLwGa==
    -----END PRIVATE KEY-----

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: "123455380079"
  location: europe-west6
  apiKey:
    type: service_account
    project_id: my-gcnv-project
    client_email: myproject-prod@my-gcnv-
project.iam.gserviceaccount.com
    client_id: "103346282737811234567"
```

```
    auth_uri: https://accounts.google.com/o/oauth2/auth
    token_uri: https://oauth2.googleapis.com/token
    auth_provider_x509_cert_url:
https://www.googleapis.com/oauth2/v1/certs
    client_x509_cert_url:
https://www.googleapis.com/robot/v1/metadata/x509/myproject-prod%40my-
gcnv-project.iam.gserviceaccount.com
  credentials:
    name: backend-tbc-gcnv-secret
  defaults:
    snapshotReserve: "10"
    exportRule: 10.0.0.0/24
  storage:
    - labels:
        performance: extreme
      serviceLevel: extreme
      defaults:
        snapshotReserve: "5"
        exportRule: 0.0.0.0/0
    - labels:
        performance: premium
      serviceLevel: premium
    - labels:
        performance: standard
      serviceLevel: standard
```

Identidade na nuvem para GKE

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-gcp-gcnv
spec:
  version: 1
  storageDriverName: google-cloud-netapp-volumes
  projectNumber: '012345678901'
  network: gcnv-network
  location: us-west2
  serviceLevel: Premium
  storagePool: pool-premium1
```

Configuração de topologias suportadas

Trident facilita o provisionamento de volumes para cargas de trabalho com base em regiões e zonas de disponibilidade. O `supportedTopologies` bloco nesta configuração de backend é usado para fornecer uma lista de regiões e zonas por backend. Os valores de região e zona especificados aqui devem corresponder aos valores de região e zona dos rótulos em cada nó de cluster Kubernetes. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em uma classe de armazenamento. Para classes de armazenamento que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria volumes na região e zona mencionadas. Para mais informações, consulte "Usar a topologia CSI".

```
---
version: 1
storageDriverName: google-cloud-netapp-volumes
subscriptionID: 9f87c765-4774-fake-ae98-a721add45451
tenantID: 68e4f836-edc1-fake-bff9-b2d865ee56cf
clientID: dd043f63-bf8e-fake-8076-8de91e5713aa
clientSecret: SECRET
location: asia-east1
serviceLevel: flex
supportedTopologies:
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-a
  - topology.kubernetes.io/region: asia-east1
    topology.kubernetes.io/zone: asia-east1-b
```

Qual é o próximo passo?

Após criar o arquivo de configuração de backend, execute o seguinte comando:

```
kubectl create -f <backend-file>
```

Para verificar se o backend foi criado com sucesso, execute o seguinte comando:

```
kubectl get tridentbackendconfig

NAME                BACKEND NAME        BACKEND UUID
PHASE   STATUS
backend-tbc-gcnv    backend-tbc-gcnv    b2fd1ff9-b234-477e-88fd-713913294f65
Bound   Success
```

Se a criação do backend falhar, há algo errado com a configuração do backend. Você pode descrever o backend usando o `kubectl get tridentbackendconfig <backend-name>` comando ou visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode excluir o backend e executar o comando create novamente.

Definições de classe de armazenamento

A seguir está uma definição básica `StorageClass` que se refere ao backend acima.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-nfs-sc
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
```

Exemplo de definições usando o `parameter.selector` campo:

Usando `parameter.selector` você pode especificar para cada `StorageClass` o "pool virtual" que é usado para hospedar um volume. O volume terá os aspectos definidos no pool escolhido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: extreme-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=extreme
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: premium-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=premium
  backendType: google-cloud-netapp-volumes

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: standard-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=standard
  backendType: google-cloud-netapp-volumes
```

Para mais detalhes sobre classes de armazenamento, consulte "Crie uma storage class".

Exemplos de definições para volumes SMB

Usando `nasType`, `node-stage-secret-name` e `node-stage-secret-namespace`, você pode especificar um volume SMB e fornecer as credenciais necessárias do Active Directory. Qualquer usuário/senha do Active Directory com qualquer/nenhuma permissão pode ser usado para o segredo do estágio do nó.

Configuração básica no namespace padrão

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: "default"
```

Usando segredos diferentes por namespace

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: "smbcreds"
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

Utilizando segredos diferentes em cada volume

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: gcnv-sc-smb
provisioner: csi.trident.netapp.io
parameters:
  backendType: "google-cloud-netapp-volumes"
  trident.netapp.io/nasType: "smb"
  csi.storage.k8s.io/node-stage-secret-name: ${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

`nasType: smb`Filtros para pools que suportam volumes SMB. `nasType: nfs ou `nasType: null`filtros para pools NFS.

Exemplo de definição de PVC

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: gcnv-nfs-pvc
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 100Gi
  storageClassName: gcnv-nfs-sc
```

Para verificar se o PVC está vinculado, execute o seguinte comando:

```
kubectl get pvc gcnv-nfs-pvc

NAME          STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS AGE
gcnv-nfs-pvc  Bound    pvc-b00f2414-e229-40e6-9b16-ee03eb79a213  100Gi
RWX            gcnv-nfs-sc  1m
```

Configurar um NetApp HCI ou SolidFire backend

Aprenda como criar e usar um backend Element com sua instalação do Trident.

Detalhes do driver Element

Trident fornece o `solidfire-san` storage driver para se comunicar com o cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

O `solidfire-san` driver de armazenamento suporta os modos de volume *file* e *block*. Para o `Filesystem` volumeMode, Trident cria um volume e cria um sistema de arquivos. O tipo de sistema de arquivos é especificado pelo StorageClass.

Driver	Protocolo	VolumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`solidfire-san`	iSCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos. Dispositivo de bloco bruto.
`solidfire-san`	iSCSI	Sistema de arquivos	RWO, RWOP	`xfs`, `ext3`, `ext4`

Antes de começar

Você precisará do seguinte antes de criar um backend Element.

- Um sistema de storage compatível que execute o software Element.
- Credenciais para um administrador de cluster NetApp HCI/SolidFire ou usuário de locatário que possa gerenciar volumes.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. Consulte "Informações sobre preparação do nó de trabalho".

Opções de configuração do backend

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	Sempre "solidfire-san"
`backendName`	Nome personalizado ou o storage backend	"solidfire_" + endereço IP de storage (iSCSI)
`Endpoint`	MVIP para o SolidFire cluster com credenciais de locatário	
`SVIP`	Endereço IP e porta de storage (iSCSI)	
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes.	""
`TenantName`	Nome do locatário a ser usado (criado se não for encontrado)	
`InitiatorIFace`	Restringir o tráfego iSCSI a uma interface de host específica	"default"
`UseCHAP`	Use CHAP para autenticar iSCSI. Trident usa CHAP.	verdadeiro
`AccessGroups`	Lista de IDs de grupos de acesso a serem usados	Encontra o ID de um grupo de acesso chamado "trident"
`Types`	Especificações de QoS	

Parâmetro	Descrição	Padrão
`limitVolumeSize`	Falhar no provisionamento se o tamanho do volume solicitado for superior a este valor	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração a serem usados na resolução de problemas. Exemplo, {"api":false, "method":true}	null

Não utilize `debugTraceFlags` a menos que esteja solucionando problemas e necessite de um despejo de logs detalhado.

Exemplo 1: configuração de backend para `solidfire-san` driver com três tipos de volume

Este exemplo mostra um arquivo de backend usando autenticação CHAP e modelando três tipos de volume com garantias de QoS específicas. Muito provavelmente, você definiria classes de armazenamento para consumir cada um deles usando o `IOPS` parâmetro de classe de armazenamento.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
labels:
  k8scluster: dev1
  backend: dev1-element-cluster
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
```

Exemplo 2: configuração de backend e classe de armazenamento para `solidfire-san`driver com pools virtuais

Este exemplo mostra o arquivo de definição de backend configurado com pools virtuais juntamente com StorageClasses que fazem referência a eles.

Trident copia os rótulos presentes em um pool de storage para o LUN de storage de backend durante o provisionamento. Para conveniência, os administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

No arquivo de definição de backend de exemplo mostrado abaixo, valores padrão específicos são definidos para todos os pools de storage, que definem o `type` em Silver. Os pools virtuais são definidos na seção `storage`. Neste exemplo, alguns pools de storage definem seu próprio tipo e alguns pools substituem os valores padrão definidos acima.

```
---
version: 1
storageDriverName: solidfire-san
Endpoint: https://<user>:<password>@<mvip>/json-rpc/8.0
SVIP: <svip>:3260
TenantName: <tenant>
UseCHAP: true
Types:
  - Type: Bronze
    Qos:
      minIOPS: 1000
      maxIOPS: 2000
      burstIOPS: 4000
  - Type: Silver
    Qos:
      minIOPS: 4000
      maxIOPS: 6000
      burstIOPS: 8000
  - Type: Gold
    Qos:
      minIOPS: 6000
      maxIOPS: 8000
      burstIOPS: 10000
type: Silver
labels:
  store: solidfire
  k8scluster: dev-1-cluster
region: us-east-1
storage:
  - labels:
      performance: gold
      cost: "4"
    zone: us-east-1a
```

```
    type: Gold
  - labels:
      performance: silver
      cost: "3"
    zone: us-east-1b
    type: Silver
  - labels:
      performance: bronze
      cost: "2"
    zone: us-east-1c
    type: Bronze
  - labels:
      performance: silver
      cost: "1"
    zone: us-east-1d
```

As seguintes definições de StorageClass referem-se aos pools virtuais acima. Usando o campo `parameters.selector`, cada StorageClass indica qual(is) pool(s) virtual(is) pode(m) ser usado(s) para hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

O primeiro StorageClass (`solidfire-gold-four` mapeará o primeiro pool virtual. Este é o único pool que oferece desempenho Gold com um `Volume Type QoS` de Gold. O último StorageClass (`solidfire-silver` indica qualquer pool de storage que ofereça desempenho Silver. Trident decidirá qual pool virtual será selecionado e garantirá que o requisito de storage seja atendido.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-gold-four
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=gold; cost=4
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-three
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=3
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
```

```
kind: StorageClass
metadata:
  name: solidfire-bronze-two
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=bronze; cost=2
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver-one
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver; cost=1
  fsType: ext4

---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: solidfire-silver
provisioner: csi.trident.netapp.io
parameters:
  selector: performance=silver
  fsType: ext4
```

Encontre mais informações

- "Grupos de acesso a volume"

Drivers SAN do ONTAP

Visão geral do driver ONTAP SAN

Saiba mais sobre como configurar um backend ONTAP com os drivers SAN do ONTAP e do Cloud Volumes ONTAP.

Detalhes do driver ONTAP SAN

Trident fornece os seguintes drivers de armazenamento SAN para se comunicar com o ONTAP cluster. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-san`	iSCSI SCSI sobre FC	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	iSCSI SCSI sobre FC	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`
`ontap-san`	NVMe/TCP Consulte Considerações adicionais para NVMe/TCP.	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san`	NVMe/TCP Consulte Considerações adicionais para NVMe/TCP.	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`
`ontap-san-economy`	iSCSI	Bloco	RWO, ROX, RWX, RWOP	Sem sistema de arquivos; dispositivo de bloco bruto
`ontap-san-economy`	iSCSI	Sistema de arquivos	RWO, RWOP ROX e RWX não estão disponíveis no modo de volume de sistema de arquivos.	`xfs`, `ext3`, `ext4`

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados" e o driver `ontap-san-economy` não puder ser usado.
- Não utilize `ontap-nas-economy` se você prevê a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- NetApp não recomenda usar o crescimento automático do FlexVol em todos os drivers ONTAP, exceto ontap-san. Como alternativa, Trident oferece suporte ao uso de reserva de snapshot e dimensiona os volumes FlexVol de acordo.

Permissões do usuário

Trident espera ser executado como administrador do ONTAP ou do SVM, normalmente usando o `admin` usuário do cluster ou um `vsadmin` usuário do SVM, ou um usuário com um nome diferente que tenha a mesma função. Para implantações do Amazon FSx for NetApp ONTAP, Trident espera ser executado como administrador do ONTAP ou do SVM, usando o usuário do cluster `fsxadmin` ou um usuário do SVM `vsadmin`, ou um usuário com um nome diferente que tenha a mesma função. O `fsxadmin` usuário é um substituto limitado para o usuário administrador do cluster.

Se você usar o `limitAggregateUsage`parâmetro, são necessárias permissões de administrador do cluster. Ao usar Amazon FSx for NetApp ONTAP com Trident, o `limitAggregateUsage`parâmetro não funcionará com as contas de usuário `vsadmin`e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva dentro do ONTAP que um driver Trident possa usar, não recomendamos isso. A maioria das novas versões do Trident chamará APIs adicionais que precisariam ser consideradas, dificultando as atualizações e tornando-as propensas a erros.

Considerações adicionais para NVMe/TCP

Trident suporta o protocolo non-volatile memory express (NVMe) usando o `ontap-san` driver incluindo:

- IPv6
- Instantâneos e clones de volumes NVMe
- Redimensionando um volume NVMe
- Importando um volume NVMe criado fora do Trident para que seu ciclo de vida possa ser gerenciado pelo Trident
- Multipathing nativo NVMe
- Encerramento correto ou incorreto dos nós K8s (24.06)

Trident não suporta:

- DH-HMAC-CHAP que é suportado nativamente pelo NVMe
- Multipathing do device mapper (DM)
- LUKS criptografia

O NVMe é suportado apenas com as APIs REST do ONTAP e não é suportado com ONTAPI (ZAPI).

Prepare-se para configurar o backend com os drivers ONTAP SAN

Compreenda os requisitos e as opções de autenticação para configurar um backend ONTAP com drivers ONTAP SAN.

Requisitos

Para todos os backends ONTAP, Trident exige que pelo menos um agregado seja atribuído à SVM.

"Sistemas ASA r2" diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Nos sistemas ASA r2, zonas de disponibilidade de storage são usadas em vez de agregados. Consulte o "este" artigo da Knowledge Base sobre como atribuir agregados a SVMs em sistemas ASA r2.

Lembre-se de que você também pode executar mais de um driver e criar classes de armazenamento que apontem para um ou outro. Por exemplo, você pode configurar uma `san-dev` classe que usa o `ontap-san` driver e uma `san-default` classe que usa o `ontap-san-economy` driver.

Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas iSCSI apropriadas instaladas. Consulte "Prepare o nó de trabalho" para obter detalhes.

Autenticar o backend ONTAP

Trident oferece dois modos de autenticação de um backend ONTAP.

- Baseado em credenciais: o nome de usuário e a senha de um usuário do ONTAP com as permissões necessárias. Recomenda-se o uso de uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Com base em certificado: Trident também pode se comunicar com um ONTAP cluster usando um certificado instalado no backend. Nesse caso, a definição do backend deve conter os valores codificados em Base64 do certificado do cliente, da chave e do certificado da CA confiável, se utilizado (recomendado).

Você pode atualizar os backends existentes para alternar entre métodos baseados em credenciais e baseados em certificados. No entanto, apenas um método de autenticação é suportado por vez. Para mudar para um método de autenticação diferente, você deve remover o método existente da configuração do backend.

Se você tentar fornecer **tanto credenciais quanto certificados**, a criação do backend falhará com um erro informando que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ativar autenticação baseada em credenciais

Trident requer as credenciais de um administrador com escopo de SVM/cluster para se comunicar com o backend do ONTAP. Recomenda-se o uso de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante a compatibilidade futura com versões do ONTAP que possam expor APIs de recursos a serem usadas por versões futuras do Trident. Uma função de login de segurança personalizada pode ser criada e usada com Trident, mas não é recomendada.

Uma definição de backend de exemplo será semelhante a esta:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: password
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password"
}
```

Lembre-se de que a definição do backend é o único local onde as credenciais são armazenadas em texto simples. Após a criação do backend, nomes de usuário/senhas são codificados em Base64 e armazenados como segredos do Kubernetes. A criação ou atualização de um backend é a única etapa que requer conhecimento das credenciais. Assim, trata-se de uma operação exclusiva do administrador, a ser realizada pelo administrador de storage do Kubernetes.

Ativar autenticação baseada em certificado

Novos e existentes backends podem usar um certificado e se comunicar com o backend do ONTAP. Três parâmetros são necessários na definição do backend.

- clientCertificate: Valor codificado em Base64 do certificado do cliente.
- clientPrivateKey: Valor codificado em Base64 da chave privada associada.
- trustedCACertificate: valor codificado em Base64 do certificado CA confiável. Se uma CA confiável estiver sendo usada, este parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as seguintes etapas.

Passos

1. Gere um certificado de cliente e uma chave. Ao gerar, defina o Nome Comum (CN) para o usuário ONTAP que será usado para autenticação.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=admin"
```

2. Adicione um certificado de CA confiável ao cluster ONTAP. Isso pode já ter sido configurado pelo administrador de storage. Ignore se nenhuma CA confiável for utilizada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado do cliente e a chave (do passo 1) no cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

Após executar este comando, ONTAP solicitará a entrada do certificado. Cole o conteúdo do `k8senv.pem` arquivo gerado na etapa 1, depois insira `END` para concluir a instalação.

4. Confirme se a função de login de segurança do ONTAP suporta `cert` método de autenticação.

```
security login create -user-or-group-name admin -application ontapi
-authentication-method cert
security login create -user-or-group-name admin -application http
-authentication-method cert
```

5. Teste a autenticação usando o certificado gerado. Substitua <ONTAP Management LIF> e <vserver name> pelo endereço IP da Management LIF e pelo nome da SVM.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifique o certificado, a chave e o certificado da CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie o backend usando os valores obtidos na etapa anterior.

```
cat cert-backend.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"trustedCACertificate": "QNFinfO...SiqOyN",
"storagePrefix": "myPrefix_"
}

tridentctl create backend -f cert-backend.json -n trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       0 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualize os métodos de autenticação ou altere as credenciais

Você pode atualizar um backend existente para usar um método de autenticação diferente ou para rotacionar suas credenciais. Isso funciona nos dois sentidos: backends que utilizam nome de usuário/senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para usar nome de usuário/senha. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl backend update`.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-san",
"backendName": "SanBackend",
"managementLIF": "1.2.3.4",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend SanBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| SanBackend | ontap-san      | 586b1cd5-8cf8-428d-a76c-2872713612c1 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Ao rotacionar senhas, o administrador de storage deve primeiro atualizar a senha do usuário no ONTAP. Em seguida, é feita uma atualização no backend. Ao rotacionar certificados, vários certificados podem ser adicionados ao usuário. O backend é então atualizado para usar o novo certificado, após o que o certificado antigo pode ser excluído do cluster ONTAP.

A atualização do backend não interrompe o acesso aos volumes já criados, nem afeta as conexões de volume feitas posteriormente. Uma atualização bem-sucedida do backend indica que Trident pode se comunicar com o ONTAP backend e lidar com operações de volume futuras.

Criar função ONTAP personalizada para Trident

Você pode criar uma função de cluster ONTAP com privilégios mínimos para que não precise usar a função de administrador do ONTAP para executar operações no Trident. Ao incluir o nome de usuário em um arquivo de configuração de backend do Trident, o Trident usa a função de cluster ONTAP que você criou para executar as operações.

Consulte "Gerador de funções personalizadas Trident" para obter mais informações sobre como criar funções personalizadas do Trident.

Usando ONTAP CLI

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Execute as seguintes etapas no ONTAP System Manager:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) Para criar uma função personalizada no nível da SVM, selecione **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **Users and Roles**.
   c. Selecione **+Adicionar** em **Roles**.
   d. Defina as regras para a função e clique em **Save**.
2. **Mapeie a função ao usuário Trident**: + Execute as seguintes etapas na página **Usuários e Funções**:
   a. Selecione o ícone Adicionar **+** em **Usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **Função**.
   c. Clique em **Salvar**.

Consulte as seguintes páginas para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Autentique conexões com CHAP bidirecional

Trident pode autenticar sessões iSCSI com CHAP bidirecional para os `ontap-san` e `ontap-san-economy` drivers. Isso requer a ativação da opção `useCHAP` na definição do seu backend. Quando definido como `true`, Trident configura a segurança do iniciador padrão da SVM para CHAP bidirecional e define o nome de usuário e os segredos do arquivo de backend. NetApp recomenda o uso de CHAP bidirecional para autenticar conexões. Veja o exemplo de configuração a seguir:

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap_san_chap
managementLIF: 192.168.0.135
svm: ontap_iscsi_svm
useCHAP: true
username: vsadmin
password: password
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
```

O `useCHAP`parâmetro é uma opção booleana que pode ser configurada apenas uma vez. Ele é definido como falso por padrão. Depois de defini-lo como verdadeiro, você não pode alterá-lo para falso.

Além de `useCHAP=true`, os campos `chapInitiatorSecret`, `chapTargetInitiatorSecret`, `chapTargetUsername` e `chapUsername` devem ser incluídos na definição do backend. Os segredos podem ser alterados após a criação de um backend executando `tridentctl update`.

Como funciona

Ao definir `useCHAP` como verdadeiro, o administrador de storage instrui Trident a configurar CHAP no backend de storage. Isso inclui o seguinte:

- Configurando CHAP no SVM:
  - Se o tipo de segurança do iniciador padrão da SVM for none (definido por padrão) **e** não houver LUNs preexistentes no volume, Trident definirá o tipo de segurança padrão para `CHAP` e prosseguirá com a configuração do nome de usuário e segredos do iniciador e destino CHAP.
  - Se a SVM contiver LUNs, Trident não habilitará CHAP na SVM. Isso garante que o acesso às LUNs já presentes na SVM não seja restringido.
- Configuração do nome de usuário e dos segredos do iniciador e do alvo CHAP; essas opções devem ser especificadas na configuração do backend (como mostrado acima).

Após a criação do backend, Trident cria um correspondente `tridentbackend` CRD e armazena os segredos CHAP e os nomes de usuário como segredos do Kubernetes. Todos os PVs que são criados pelo Trident nesse backend serão montados e anexados via CHAP.

Rotacionar credenciais e atualizar backends

Você pode atualizar as credenciais CHAP atualizando os parâmetros CHAP no `backend.json` arquivo. Isso exigirá atualizar os segredos CHAP e usar o `tridentctl update` comando para refletir essas alterações.

Ao atualizar os segredos CHAP de um backend, você deve usar `tridentctl` para atualizar o backend. Não atualize as credenciais no cluster de storage usando a ONTAP CLI ou ONTAP System Manager, pois Trident não conseguirá detectar essas alterações.

```
cat backend-san.json
{
    "version": 1,
    "storageDriverName": "ontap-san",
    "backendName": "ontap_san_chap",
    "managementLIF": "192.168.0.135",
    "svm": "ontap_iscsi_svm",
    "useCHAP": true,
    "username": "vsadmin",
    "password": "password",
    "chapInitiatorSecret": "cl9qxUpDaTeD",
    "chapTargetInitiatorSecret": "rqxigXgkeUpDaTeD",
    "chapTargetUsername": "iJF4heBRT0TCwxyz",
    "chapUsername": "uh2aNCLSd6cNwxyz",
}

./tridentctl update backend ontap_san_chap -f backend-san.json -n trident
+----------------+----------------+--------------------------------------
+--------+---------+
|   NAME         | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+----------------+----------------+--------------------------------------
+--------+---------+
| ontap_san_chap | ontap-san      | aa458f3b-ad2d-4378-8a33-1a472ffbeb5c |
online |       7 |
+----------------+----------------+--------------------------------------
+--------+---------+
```

As conexões existentes permanecerão inalteradas; elas continuarão ativas se as credenciais forem atualizadas pelo Trident no SVM. Novas conexões usam as credenciais atualizadas e as conexões existentes continuam ativas. Desconectar e reconectar PVs antigos fará com que eles usem as credenciais atualizadas.

Opções e exemplos de configuração do ONTAP SAN

Aprenda como criar e usar drivers ONTAP SAN com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses.

"Sistemas ASA r2" diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage. Essas variações impactam o uso de certos parâmetros conforme indicado. "Saiba mais sobre as diferenças entre sistemas ASA r2 e outros sistemas ONTAP".

Apenas o `ontap-san`driver (com os protocolos iSCSI, NVMe/TCP e FC) é compatível com sistemas ASA r2.

Na configuração do backend do Trident, não é necessário especificar que seu sistema é ASA r2. Ao selecionar `ontap-san` como o `storageDriverName`, o Trident detecta automaticamente os sistemas ASA r2 ou outros sistemas ONTAP. Alguns parâmetros de configuração do backend não se aplicam a sistemas ASA r2, conforme indicado na tabela abaixo.

Opções de configuração do backend

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	`ontap-san` ou `ontap-san-economy`
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. É possível especificar um nome de domínio totalmente qualificado (FQDN). Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando a flag IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para um switchover MetroCluster perfeito, consulte o Exemplo do MetroCluster. Se estiver usando credenciais "vsadmin", `managementLIF` deve ser a da SVM; se estiver usando credenciais "admin", `managementLIF` deve ser a do cluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o parâmetro IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Não especifique para iSCSI.** Trident usa "Mapa de LUN seletivo do ONTAP" para descobrir os LIFs iSCSI necessários para estabelecer uma sessão de múltiplos caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. **Omita para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado pelo SVM

Parâmetro	Descrição	Padrão
`svm`	Máquina virtual de storage a ser usada **Omitir para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`useCHAP`	Use CHAP para autenticar iSCSI para drivers ONTAP SAN [parâmetro booleano]. Defina como `true` para que Trident configure e use CHAP bidirecional como autenticação padrão para o SVM fornecido no backend. Consulte "Prepare-se para configurar o backend com os drivers ONTAP SAN" para detalhes. **Não compatível com FCP ou NVMe/TCP.**	`false`
`chapInitiatorSecret`	Segredo do iniciador CHAP. Obrigatório se `useCHAP=true`	""
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`chapTargetInitiatorSecret`	Segredo do iniciador do alvo CHAP. Obrigatório se `useCHAP=true`	""
`chapUsername`	Nome de usuário de entrada. Obrigatório se `useCHAP=true`	""
`chapTargetUsername`	Nome de usuário de destino. Obrigatório se `useCHAP=true`	""
`clientCertificate`	Valor codificado em Base64 do certificado do cliente. Usado para autenticação baseada em certificado	""
`clientPrivateKey`	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em Base64 do certificado da CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário necessário para se comunicar com o cluster ONTAP. Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	""
`password`	Senha necessária para se comunicar com o cluster ONTAP. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	""
`svm`	Máquina virtual de storage para usar	Derivado se uma SVM `managementLIF` for especificada
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser modificado posteriormente. Para atualizar este parâmetro, você precisará criar um novo backend.	`trident`

Parâmetro	Descrição	Padrão
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend entrará em estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especifique para sistemas ASA r2**.	""
`limitAggregateUsage`	O provisionamento falhará se o uso ultrapassar essa porcentagem. Se você estiver usando um Amazon FSx para NetApp ONTAP backend, não especifique `limitAggregateUsage`. Os `fsxadmin` e `vsadmin` fornecidos não contêm as permissões necessárias para recuperar o uso agregado e limitá-lo usando Trident. **Não especifique para sistemas ASA r2**.	"" (não aplicado por padrão)
`limitVolumeSize`	O provisionamento falha se o tamanho do volume solicitado for superior a este valor. Também restringe o tamanho máximo dos volumes que gerencia para LUNs.	"" (não aplicado por padrão)
`lunsPerFlexvol`	Número máximo de LUNs por FlexVol, deve estar no intervalo [50, 200]	`100`
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} Não use a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	`null`

Parâmetro	Descrição	Padrão
`useREST`	Parâmetro booleano para usar as APIs REST do ONTAP. `useREST`Quando definido como `true`, Trident usa as APIs REST do ONTAP para se comunicar com o backend; quando definido como `false`, Trident usa chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 ou posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao aplicativo `ontapi`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`. A partir da versão 24.06 do Trident e ONTAP 9.15.1 ou posterior, `useREST`é definido como `true` por padrão; altere `useREST` para `false` para usar chamadas ONTAPI (ZAPI). `useREST` está totalmente qualificado para NVMe/TCP. O NVMe é suportado apenas com as APIs REST do ONTAP e não é suportado com ONTAPI (ZAPI). **Se especificado, defina sempre como `true` para sistemas ASA r2**.	`true`para ONTAP 9.15.1 ou posterior, caso contrário `false`.
`sanType`	Use para selecionar `iscsi` para iSCSI, `nvme` para NVMe/TCP ou `fcp` para SCSI sobre Fibre Channel (FC).	`iscsi` se estiver em branco

Parâmetro	Descrição	Padrão
`formatOptions`	Use `formatOptions` para especificar argumentos de linha de comando para o comando `mkfs`, que serão aplicados sempre que um volume for formatado. Isso permite formatar o volume de acordo com suas preferências. Certifique-se de especificar as formatOptions de forma semelhante às opções do comando mkfs, excluindo o caminho do dispositivo. Exemplo: "-E nodiscard" **Compatível para `ontap-san` e `ontap-san-economy`drivers com o protocolo iSCSI. Além disso, compatível com sistemas ASA r2 ao usar os protocolos iSCSI e NVMe/TCP.**	
`limitVolumePoolSize`	Tamanho máximo solicitável de FlexVol ao usar LUNs no backend ontap-san-economy.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe `ontap-san-economy`os backends de criar novos volumes FlexVol para conter seus LUNs. Somente FlexVols preexistentes são usados para provisionar novos PVs.	

Recomendações para uso de formatOptions

Trident recomenda as seguintes opções para agilizar o processo de formatação:

- **-E nodiscard (ext3, ext4):** Não tente descartar blocos durante a criação do sistema de arquivos (o descarte inicial de blocos é útil em dispositivos de estado sólido e em storage esparso/com thin provisioning). Esta opção substitui a opção obsoleta "-K" e é aplicável aos sistemas de arquivos ext3 e ext4.
- **-K (xfs):** Não tente descartar blocos durante a criação do sistema de arquivos (mkfs). Esta opção se aplica ao sistema de arquivos xfs.

Autentique o Trident em um SVM de backend usando credenciais do Active Directory

Você pode configurar Trident para autenticar em uma SVM de backend usando credenciais do Active Directory (AD). Antes que uma conta do AD possa acessar a SVM, você deve configurar o acesso do controlador de domínio do AD ao cluster ou à SVM. Para administração do cluster com uma conta do AD, você deve criar domain tunnel. Consulte "Configurar o acesso do controlador de domínio do Active Directory no ONTAP" para obter detalhes.

passos

1. Configurar as definições do Domain Name System (DNS) para uma SVM de backend:

```
vserver services dns create -vserver <svm_name> -dns-servers
<dns_server_ip1>,<dns_server_ip2>
```

2. Execute o seguinte comando para criar uma conta de computador para a SVM no Active Directory:

```
vserver active-directory create -vserver DataSVM -account-name ADSERVER1
-domain demo.netapp.com
```

3. Use este comando para criar um usuário ou grupo do AD para gerenciar o cluster ou SVM

```
security login create -vserver <svm_name> -user-or-group-name
<ad_user_or_group> -application <application> -authentication-method domain
-role vsadmin
```

4. No arquivo de configuração do Trident backend, defina os parâmetros `username` e `password` para o nome de usuário ou grupo do AD e a senha, respectivamente.

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	"verdadeiro" **Se especificado, defina como `true` para sistemas ASA r2**.
`spaceReserve`	Modo de reserva de espaço; "nenhum" (com thin provisioning) ou "volume" (thick). **Definido como `none` para sistemas ASA r2**.	"none"
`snapshotPolicy`	Política do Snapshot a ser usada. **Definida como `none` para sistemas ASA r2**.	"none"
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha um dos qosPolicy ou adaptiveQosPolicy por pool de storage/backend. O uso de grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o limite máximo para a taxa de transferência total de todas as cargas de trabalho.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a ser atribuído aos volumes criados. Escolha uma de qosPolicy ou adaptiveQosPolicy por pool de storage/backend	""
`snapshotReserve`	Percentual do volume reservado para snapshots. **Não especificar para sistemas ASA r2**.	"0" se `snapshotPolicy` for "none", caso contrário ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	"false"
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	"falso" **Se especificado, defina como `true` para sistemas ASA r2**.

Parâmetro	Descrição	Padrão
`luksEncryption`	Ative a criptografia LUKS. Consulte "Use Linux Unified Key Setup (LUKS)".	**Defina como `false` para sistemas ASA r2.**
`tieringPolicy`	Política de tiering para usar "none" **Não especificar para sistemas ASA r2.**	
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

Exemplos de provisionamento de volume

Aqui está um exemplo com valores padrão definidos:

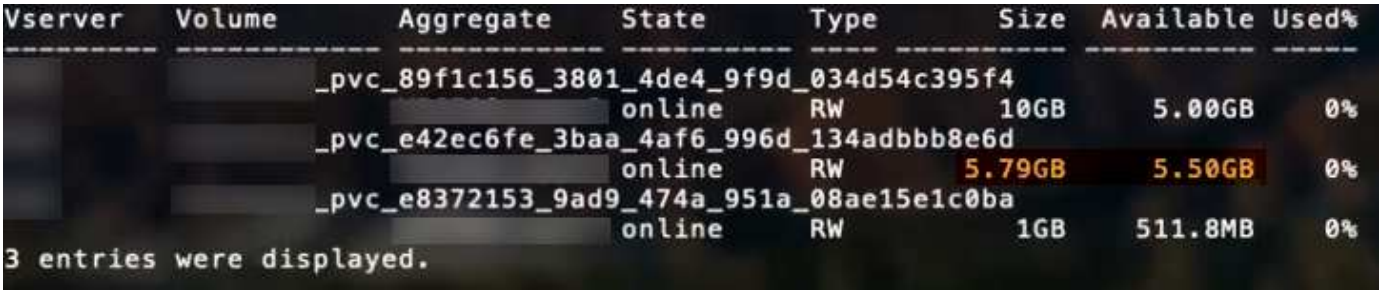
```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: trident_svm
username: admin
password: <password>
labels:
  k8scluster: dev2
  backend: dev2-sanbackend
storagePrefix: alternate-trident
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: standard
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

> Para todos os volumes criados usando o `ontap-san` driver, Trident adiciona 10 por cento de capacidade extra ao FlexVol para acomodar os metadados do LUN. O LUN será provisionado com o tamanho exato que o usuário solicitar no PVC. Trident adiciona 10 por cento ao FlexVol (exibido como tamanho disponível no ONTAP). Os usuários agora receberão a quantidade de capacidade utilizável que solicitaram. Essa alteração também impede que os LUNs se tornem somente leitura, a menos que o espaço disponível esteja totalmente utilizado. Isso não se aplica ao ontap-san-economy.

Para backends que definem `snapshotReserve`, Trident calcula o tamanho dos volumes da seguinte forma:

```
Total volume size = [(PVC requested size) / (1 - (snapshotReserve
percentage) / 100)] * 1.1
```

O valor 1.1 representa os 10% adicionais que o Trident adiciona ao FlexVol para acomodar os metadados da LUN. Para `snapshotReserve` = 5%, e solicitação de PVC = 5 GiB, o tamanho total do volume é 5,79 GiB e o tamanho disponível é 5,5 GiB. O comando `volume show` deve exibir resultados semelhantes a este exemplo:



Atualmente, o redimensionamento é a única maneira de usar o novo cálculo para um existing volume.

Exemplos de configuração mínima

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros com os valores padrão. Esta é a maneira mais fácil de definir um backend.

> Se você estiver usando Amazon FSx no NetApp ONTAP com Trident, NetApp recomenda que você especifique nomes DNS para LIFs em vez de endereços IP.

Exemplo de ONTAP SAN

Esta é uma configuração básica usando o `ontap-san` driver.

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
username: vsadmin
password: <password>
```

Exemplo do MetroCluster

Você pode configurar o backend para evitar ter que atualizar manualmente a definição do backend após switchover e switchback durante "Replicação e recuperação de SVM".

Para switchover e switchback sem interrupções, especifique a SVM usando `managementLIF` e omita os parâmetros `svm`. Por exemplo:

```
version: 1
storageDriverName: ontap-san
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemplo de economia ONTAP SAN

```
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
username: vsadmin
password: <password>
```

Exemplo de autenticação baseada em certificado

Neste exemplo de configuração básica `clientCertificate`, `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando uma CA confiável) são preenchidos em `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado da CA confiável, respectivamente.

```
---
version: 1
storageDriverName: ontap-san
backendName: DefaultSANBackend
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
```

Exemplos bidirecionais CHAP

Esses exemplos criam um backend com `useCHAP` definido como `true`.

Exemplo ONTAP SAN CHAP

```
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
labels:
  k8scluster: test-cluster-1
  backend: testcluster1-sanbackend
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de economia SAN CHAP do ONTAP

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
```

Exemplo de NVMe/TCP

Você precisa ter uma SVM configurada com NVMe no seu backend ONTAP. Esta é uma configuração básica de backend para NVMe/TCP.

```
---
version: 1
backendName: NVMeBackend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_nvme
username: vsadmin
password: password
sanType: nvme
useREST: true
```

Exemplo de SCSI sobre FC (FCP)

Você precisa ter uma SVM configurada com FC no seu ONTAP backend. Esta é uma configuração básica de backend para FC.

```
---
version: 1
backendName: fcp-backend
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_fc
username: vsadmin
password: password
sanType: fcp
useREST: true
```

Exemplo de configuração de backend com nameTemplate

```
---
version: 1
storageDriverName: ontap-san
backendName: ontap-san-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

formatOptions exemplo para o driver ontap-san-economy

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: ""
svm: svm1
username: ""
password: "!"
storagePrefix: whelk_
debugTraceFlags:
  method: true
  api: true
defaults:
  formatOptions: -E nodiscard
```

Exemplos de backends com pools virtuais

Nesses arquivos de definição de backend de exemplo, valores padrão específicos são definidos para todos os pools de storage, como `spaceReserve` em none, `spaceAllocation` em false e `encryption` em false. Os pools virtuais são definidos na seção de storage.

Trident define rótulos de provisionamento no campo "Comentários". Os comentários são definidos no volume FlexVol; Trident copia todos os rótulos presentes em um pool virtual para o volume de storage durante o provisionamento. Para conveniência, administradores de storage podem definir rótulos por pool virtual e agrupar volumes por rótulo.

Nestes exemplos, alguns pools de storage definem seus próprios `spaceReserve`, `spaceAllocation`, e `encryption` valores, e alguns pools substituem os valores padrão.

Exemplo de ONTAP SAN

```yaml
---
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
svm: svm_iscsi
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
  qosPolicy: standard
labels:
  store: san_store
  kubernetes-cluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "40000"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
      adaptiveQosPolicy: adaptive-extreme
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
      qosPolicy: premium
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemplo de economia ONTAP SAN

```
---
version: 1
storageDriverName: ontap-san-economy
managementLIF: 10.0.0.1
svm: svm_iscsi_eco
useCHAP: true
chapInitiatorSecret: cl9qxIm36DKyawxy
chapTargetInitiatorSecret: rqxigXgkesIpwxyz
chapTargetUsername: iJF4heBRT0TCwxyz
chapUsername: uh2aNCLSd6cNwxyz
username: vsadmin
password: <password>
defaults:
  spaceAllocation: "false"
  encryption: "false"
labels:
  store: san_economy_store
region: us_east_1
storage:
  - labels:
      app: oracledb
      cost: "30"
    zone: us_east_1a
    defaults:
      spaceAllocation: "true"
      encryption: "true"
  - labels:
      app: postgresdb
      cost: "20"
    zone: us_east_1b
    defaults:
      spaceAllocation: "false"
      encryption: "true"
  - labels:
      app: mysqldb
      cost: "10"
    zone: us_east_1c
    defaults:
      spaceAllocation: "true"
      encryption: "false"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1c
```

```
    defaults:
      spaceAllocation: "true"
      encryption: "false"
```

Exemplo de NVMe/TCP

```
---
version: 1
storageDriverName: ontap-san
sanType: nvme
managementLIF: 10.0.0.1
svm: nvme_svm
username: vsadmin
password: <password>
useREST: true
defaults:
  spaceAllocation: "false"
  encryption: "true"
storage:
  - labels:
      app: testApp
      cost: "20"
    defaults:
      spaceAllocation: "false"
      encryption: "false"
```

Mapear back-ends para StorageClasses

As seguintes definições de StorageClass referem-se ao Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass será mapeado para o primeiro pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass corresponderá ao segundo e terceiro pool virtual no `ontap-san` backend. Esses são os únicos pools que oferecem um nível de proteção diferente de gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass será mapeado para o terceiro pool virtual no `ontap-san-economy` backend. Este é o único pool que oferece configuração de pool de storage para o aplicativo do tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o segundo pool virtual no `ontap-san` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass corresponderá ao terceiro pool virtual no `ontap-san` backend e ao quarto pool virtual no `ontap-san-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

- O `my-test-app-sc` StorageClass será mapeado para o `testAPP` pool virtual no `ontap-san` driver com `sanType: nvme`. Este é o único pool que oferece `testApp`.

```
---
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: my-test-app-sc
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=testApp"
  fsType: "ext4"
```

Trident decidirá qual pool virtual será selecionado e garantirá que o requisito de storage seja atendido.

Drivers NAS do ONTAP

Visão geral do driver ONTAP NAS

Saiba mais sobre como configurar um backend ONTAP com os drivers NAS do ONTAP e do Cloud Volumes ONTAP.

Detalhes do driver ONTAP NAS

Trident fornece os seguintes drivers de armazenamento NAS para se comunicar com o cluster ONTAP. Os modos de acesso suportados são: *ReadWriteOnce* (RWO), *ReadOnlyMany* (ROX), *ReadWriteMany* (RWX), *ReadWriteOncePod* (RWOP).

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`
`ontap-nas-economy`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

Driver	Protocolo	volumeMode	Modos de acesso suportados	Sistemas de arquivos suportados
`ontap-nas-flexgroup`	NFS SMB	Sistema de arquivos	RWO, ROX, RWX, RWOP	"", `nfs`, `smb`

- Use `ontap-san-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados".
- Use `ontap-nas-economy` somente se a contagem de uso de volume persistente prevista for maior que "limites de volume ONTAP suportados" e o driver `ontap-san-economy` não puder ser usado.
- Não utilize `ontap-nas-economy` se você prevê a necessidade de proteção de dados, recuperação de desastres ou mobilidade.
- NetApp não recomenda usar o crescimento automático do FlexVol em todos os drivers ONTAP, exceto ontap-san. Como alternativa, Trident oferece suporte ao uso de reserva de snapshot e dimensiona os volumes FlexVol de acordo.

Permissões do usuário

Trident espera ser executado como administrador do ONTAP ou do SVM, normalmente usando o `admin` cluster user ou um `vsadmin` SVM user, ou um usuário com um nome diferente que tenha a mesma função.

Para implantações do Amazon FSx for NetApp ONTAP, o Trident espera ser executado como administrador do ONTAP ou do SVM, usando o usuário do cluster `fsxadmin` ou um usuário do SVM `vsadmin`, ou um usuário com um nome diferente que tenha o mesmo papel. O usuário `fsxadmin` é um substituto limitado para o usuário administrador do cluster.

Se você usar o `limitAggregateUsage`parâmetro, são necessárias permissões de administrador do cluster. Ao usar Amazon FSx for NetApp ONTAP com Trident, o `limitAggregateUsage`parâmetro não funcionará com as contas de usuário `vsadmin`e `fsxadmin`. A operação de configuração falhará se você especificar este parâmetro.

Embora seja possível criar uma função mais restritiva dentro do ONTAP que um driver Trident possa usar, não recomendamos isso. A maioria das novas versões do Trident chamará APIs adicionais que precisariam ser consideradas, dificultando as atualizações e tornando-as propensas a erros.

Prepare-se para configurar um backend com drivers ONTAP NAS

Compreenda os requisitos, as opções de autenticação e as políticas de exportação para configurar um backend ONTAP com drivers ONTAP NAS.

A partir da versão 25.10, NetApp Trident oferece suporte "NetApp AFX sistema de storage". NetApp AFX storage systems diferem de outros sistemas ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage.

Apenas o `ontap-nas`driver (com o protocolo NFS) é compatível com sistemas AFX; o protocolo SMB não é compatível.

Na configuração do backend do Trident, não é necessário especificar que seu sistema é AFX. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente os sistemas AFX.

Requisitos

- Para todos os backends ONTAP, Trident exige que pelo menos um agregado seja atribuído à SVM.
- Você pode executar mais de um driver e criar classes de armazenamento que apontem para um ou outro. Por exemplo, você pode configurar uma classe Gold que usa o driver `ontap-nas` e uma classe Bronze que usa o `ontap-nas-economy`.
- Todos os seus nós de trabalho do Kubernetes devem ter as ferramentas NFS apropriadas instaladas. Consulte "aqui" para mais detalhes.
- Trident suporta volumes SMB montados em pods executados apenas em nós Windows. Consulte Prepare-se para provisionar volumes SMB para obter detalhes.

Autenticar o backend ONTAP

Trident oferece dois modos de autenticação de um backend ONTAP.

- Baseado em credenciais: Este modo requer permissões suficientes no backend do ONTAP. Recomenda-se usar uma conta associada a uma função de login de segurança predefinida, como `admin` ou `vsadmin` para garantir a máxima compatibilidade com as versões do ONTAP.
- Baseado em certificado: Este modo requer um certificado instalado no backend para o Trident se comunicar com um cluster ONTAP. Aqui, a definição do backend deve conter os valores codificados em Base64 do certificado do cliente, da chave e do certificado da CA confiável, se utilizado (recomendado).

Você pode atualizar os backends existentes para alternar entre métodos baseados em credenciais e baseados em certificados. No entanto, apenas um método de autenticação é suportado por vez. Para mudar para um método de autenticação diferente, você deve remover o método existente da configuração do backend.

> Se você tentar fornecer **tanto credenciais quanto certificados**, a criação do backend falhará com um erro informando que mais de um método de autenticação foi fornecido no arquivo de configuração.

Ativar autenticação baseada em credenciais

Trident requer as credenciais de um administrador com escopo de SVM/cluster para se comunicar com o backend do ONTAP. Recomenda-se o uso de funções padrão predefinidas, como `admin` ou `vsadmin`. Isso garante a compatibilidade futura com versões do ONTAP que possam expor APIs de recursos a serem usadas por versões futuras do Trident. Uma função de login de segurança personalizada pode ser criada e usada com Trident, mas não é recomendada.

Uma definição de backend de exemplo será semelhante a esta:

YAML

```yaml
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
credentials:
  name: secret-backend-creds
```

JSON

```json
{
  "version": 1,
  "backendName": "ExampleBackend",
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "credentials": {
        "name": "secret-backend-creds"
    }
}
```

Lembre-se de que a definição do backend é o único local onde as credenciais são armazenadas em texto simples. Após a criação do backend, os nomes de usuário/senhas são codificados em Base64 e armazenados como segredos do Kubernetes. A criação/atualização de um backend é a única etapa que requer conhecimento das credenciais. Assim, é uma operação exclusiva do administrador, a ser realizada pelo administrador de storage do Kubernetes.

Habilitar autenticação baseada em certificado

Novos e existentes backends podem usar um certificado e se comunicar com o backend do ONTAP. Três parâmetros são necessários na definição do backend.

- clientCertificate: Valor codificado em Base64 do certificado do cliente.
- clientPrivateKey: Valor codificado em Base64 da chave privada associada.
- trustedCACertificate: valor codificado em Base64 do certificado CA confiável. Se uma CA confiável estiver sendo usada, este parâmetro deve ser fornecido. Isso pode ser ignorado se nenhuma CA confiável for usada.

Um fluxo de trabalho típico envolve as seguintes etapas.

Passos

1. Gere um certificado de cliente e uma chave. Ao gerar, defina o Nome Comum (CN) para o usuário ONTAP que será usado para autenticação.

```
openssl req -x509 -nodes -days 1095 -newkey rsa:2048 -keyout k8senv.key
-out k8senv.pem -subj "/C=US/ST=NC/L=RTP/O=NetApp/CN=vsadmin"
```

2. Adicione um certificado de CA confiável ao cluster ONTAP. Isso pode já ter sido configurado pelo administrador de storage. Ignore se nenhuma CA confiável for utilizada.

```
security certificate install -type server -cert-name <trusted-ca-cert-
name> -vserver <vserver-name>
ssl modify -vserver <vserver-name> -server-enabled true -client-enabled
true -common-name <common-name> -serial <SN-from-trusted-CA-cert> -ca
<cert-authority>
```

3. Instale o certificado do cliente e a chave (do passo 1) no cluster ONTAP.

```
security certificate install -type client-ca -cert-name <certificate-
name> -vserver <vserver-name>
security ssl modify -vserver <vserver-name> -client-enabled true
```

4. Confirme se a função de login de segurança do ONTAP suporta `cert` método de autenticação.

```
security login create -user-or-group-name vsadmin -application ontapi
-authentication-method cert -vserver <vserver-name>
security login create -user-or-group-name vsadmin -application http
-authentication-method cert -vserver <vserver-name>
```

5. Teste autenticação usando certificado gerado. Substitua <ONTAP Management LIF> e <vserver name> pelo endereço IP do Management LIF e o nome do SVM. Você deve garantir que o LIF tenha sua política de serviço definida como `default-data-management`.

```
curl -X POST -Lk https://<ONTAP-Management-
LIF>/servlets/netapp.servlets.admin.XMLrequest_filer --key k8senv.key
--cert ~/k8senv.pem -d '<?xml version="1.0" encoding="UTF-8"?><netapp
xmlns="http://www.netapp.com/filer/admin" version="1.21"
vfiler="<vserver-name>"><vserver-get></vserver-get></netapp>'
```

6. Codifique o certificado, a chave e o certificado da CA confiável com Base64.

```
base64 -w 0 k8senv.pem >> cert_base64
base64 -w 0 k8senv.key >> key_base64
base64 -w 0 trustedca.pem >> trustedca_base64
```

7. Crie o backend usando os valores obtidos na etapa anterior.

```
cat cert-backend-updated.json
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"clientCertificate": "Faaaakkkkeeee...Vaaalllluuuueeee",
"clientPrivateKey": "LS0tFaKE...0VaLuES0tLS0K",
"storagePrefix": "myPrefix_"
}

#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Atualize os métodos de autenticação ou altere as credenciais

Você pode atualizar um backend existente para usar um método de autenticação diferente ou para rotacionar suas credenciais. Isso funciona nos dois sentidos: backends que utilizam nome de usuário/senha podem ser atualizados para usar certificados; backends que utilizam certificados podem ser atualizados para usar nome de usuário/senha. Para fazer isso, você deve remover o método de autenticação existente e adicionar o novo método de autenticação. Em seguida, use o arquivo backend.json atualizado contendo os parâmetros necessários para executar `tridentctl update backend`.

```
cat cert-backend-updated.json
```

```
{
"version": 1,
"storageDriverName": "ontap-nas",
"backendName": "NasBackend",
"managementLIF": "1.2.3.4",
"dataLIF": "1.2.3.8",
"svm": "vserver_test",
"username": "vsadmin",
"password": "password",
"storagePrefix": "myPrefix_"
}
```

```
#Update backend with tridentctl
tridentctl update backend NasBackend -f cert-backend-updated.json -n
trident
+------------+----------------+--------------------------------------
+--------+---------+
|    NAME    | STORAGE DRIVER |                 UUID                 |
STATE   | VOLUMES |
+------------+----------------+--------------------------------------
+--------+---------+
| NasBackend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       9 |
+------------+----------------+--------------------------------------
+--------+---------+
```

Ao rotacionar senhas, o administrador de storage deve primeiro atualizar a senha do usuário no ONTAP. Em seguida, é feita uma atualização no backend. Ao rotacionar certificados, vários certificados podem ser adicionados ao usuário. O backend é então atualizado para usar o novo certificado, após o que o certificado antigo pode ser excluído do cluster ONTAP.

A atualização do backend não interrompe o acesso aos volumes já criados, nem afeta as conexões de volume feitas posteriormente. Uma atualização bem-sucedida do backend indica que Trident pode se comunicar com o ONTAP backend e lidar com operações de volume futuras.

Criar função ONTAP personalizada para Trident

Você pode criar uma função de cluster ONTAP com privilégios mínimos para que não precise usar a função de administrador do ONTAP para executar operações no Trident. Ao incluir o nome de usuário em um arquivo de configuração de backend do Trident, o Trident usa a função de cluster ONTAP que você criou para executar as operações.

Consulte "Gerador de funções personalizadas Trident" para obter mais informações sobre como criar funções personalizadas do Trident.

Usando ONTAP CLI

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
–vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod <password\> -role <name_of_role_in_step_1\> –vserver
<svm_name\> -comment "user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> –vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Execute as seguintes etapas no ONTAP System Manager:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) Para criar uma função personalizada no nível da SVM, selecione **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **Users and Roles**.
   c. Selecione **+Adicionar** em **Roles**.
   d. Defina as regras para a função e clique em **Save**.
2. **Mapeie a função ao usuário Trident**: + Execute as seguintes etapas na página **Usuários e Funções**:
   a. Selecione o ícone Adicionar **+** em **Usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **Função**.
   c. Clique em **Salvar**.

Consulte as seguintes páginas para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Gerenciar políticas de exportação NFS

Trident usa políticas de exportação NFS para controlar o acesso aos volumes que provisiona.

Trident oferece duas opções ao trabalhar com políticas de exportação:

- Trident pode gerenciar dinamicamente a própria política de exportação; nesse modo de operação, o

administrador de storage especifica uma lista de blocos CIDR que representam endereços IP admissíveis. Trident adiciona automaticamente os IPs de nó aplicáveis que se enquadram nesses intervalos à política de exportação no momento da publicação. Alternativamente, quando nenhum CIDR é especificado, todos os IPs unicast de escopo global encontrados no nó para o qual o volume está sendo publicado serão adicionados à política de exportação.

- Os administradores de storage podem criar uma política de exportação e adicionar regras manualmente. Trident usa a política de exportação padrão, a menos que um nome de política de exportação diferente seja especificado na configuração.

Gerencie dinamicamente as políticas de exportação

Trident oferece a capacidade de gerenciar dinamicamente políticas de exportação para backends ONTAP. Isso fornece ao administrador de storage a capacidade de especificar um espaço de endereços permitido para os endereços IP dos nós de trabalho, em vez de definir regras explícitas manualmente. Isso simplifica muito o gerenciamento de políticas de exportação; as modificações na política de exportação não exigem mais intervenção manual no cluster de storage. Além disso, isso ajuda a restringir o acesso ao cluster de storage apenas aos nós de trabalho que estão montando volumes e possuem endereços IP no intervalo especificado, oferecendo um gerenciamento detalhado e automatizado.

> Não utilize Network Address Translation (NAT) ao usar políticas de exportação dinâmicas. Com NAT, o controlador de storage vê o endereço NAT de frontend e não o endereço IP real do host, portanto, o acesso será negado quando nenhuma correspondência for encontrada nas regras de exportação.

Exemplo

Existem duas opções de configuração que devem ser usadas. Veja um exemplo de definição de backend:

```
---
version: 1
storageDriverName: ontap-nas-economy
backendName: ontap_nas_auto_export
managementLIF: 192.168.0.135
svm: svm1
username: vsadmin
password: password
autoExportCIDRs:
  - 192.168.0.0/24
autoExportPolicy: true
```

> Ao usar este recurso, você deve garantir que a junção raiz em sua SVM tenha uma política de exportação previamente criada com uma regra de exportação que permita o bloco CIDR do nó (como a política de exportação padrão). Sempre siga a prática recomendada pela NetApp de dedicar uma SVM ao Trident.

Aqui está uma explicação de como esse recurso funciona usando o exemplo acima:

- `autoExportPolicy` está definido como `true`. Isso indica que Trident cria uma política de exportação para cada volume provisionado com este backend para o `svm1` SVM e lida com a adição e exclusão de

regras usando `autoexportCIDRs` blocos de endereços. Até que um volume seja anexado a um nó, o volume usa uma política de exportação vazia, sem regras, para impedir o acesso indesejado a esse volume. Quando um volume é publicado em um nó, Trident cria uma política de exportação com o mesmo nome da qtree subjacente, contendo o endereço IP do nó dentro do bloco CIDR especificado. Esses IPs também serão adicionados à política de exportação usada pelo volume pai FlexVol.

  - Por exemplo:
    - UUID do backend 403b5326-8482-40db-96d0-d83fb3f4daec
    - `autoExportPolicy`definido para `true`
    - prefixo de storage `trident`
    - PVC UUID a79bcf5f-7b6d-4a40-9876-e2551f159c1c
    - qtree denominada trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c cria uma política de exportação para a FlexVol denominada `trident-403b5326-8482-40db96d0-d83fb3f4daec`, uma política de exportação para a qtree denominada `trident_pvc_a79bcf5f_7b6d_4a40_9876_e2551f159c1c`, e uma política de exportação vazia denominada `trident_empty` no SVM. As regras para a política de exportação da FlexVol serão um superconjunto de quaisquer regras contidas nas políticas de exportação da qtree. A política de exportação vazia será reutilizada por quaisquer volumes que não estejam anexados.

- `autoExportCIDRs` contém uma lista de blocos de endereços. Este campo é opcional e ele é definido por padrão como ["0.0.0.0/0", "::/0"]. Se não for definido, Trident adiciona todos os endereços unicast de escopo global encontrados nos nós de trabalho com publicações.

Neste exemplo, o `192.168.0.0/24` espaço de endereços é fornecido. Isso indica que os IPs dos nós do Kubernetes que se enquadram nesse intervalo de endereços com publicações serão adicionados à política de exportação que o Trident cria. Quando o Trident registra um nó no qual está sendo executado, ele recupera os endereços IP do nó e os compara com os blocos de endereços fornecidos em `autoExportCIDRs`. No momento da publicação, após filtrar os IPs, o Trident cria as regras da política de exportação para os endereços IP dos clientes do nó para o qual está publicando.

Você pode atualizar `autoExportPolicy` e `autoExportCIDRs` para backends após criá-los. Você pode adicionar novos CIDRs para um backend que é gerenciado automaticamente ou excluir CIDRs existentes. Tenha cuidado ao excluir CIDRs para garantir que conexões existentes não sejam interrompidas. Você também pode optar por desativar `autoExportPolicy` para um backend e voltar para uma política de exportação criada manualmente. Isso exigirá a configuração do parâmetro `exportPolicy` no seu arquivo de configuração do backend.

Após Trident criar ou atualizar um backend, você pode verificar o backend usando `tridentctl` ou o correspondente `tridentbackend` CRD:

```
./tridentctl get backends ontap_nas_auto_export -n trident -o yaml
items:
- backendUUID: 403b5326-8482-40db-96d0-d83fb3f4daec
  config:
    aggregate: ""
    autoExportCIDRs:
    - 192.168.0.0/24
    autoExportPolicy: true
    backendName: ontap_nas_auto_export
    chapInitiatorSecret: ""
    chapTargetInitiatorSecret: ""
    chapTargetUsername: ""
    chapUsername: ""
    dataLIF: 192.168.0.135
    debug: false
    debugTraceFlags: null
    defaults:
      encryption: "false"
      exportPolicy: <automatic>
      fileSystemType: ext4
```

Quando um nó é removido, Trident verifica todas as políticas de exportação para remover as regras de acesso correspondentes ao nó. Ao remover esse endereço IP do nó das políticas de exportação dos backends gerenciados, Trident impede montagens não autorizadas, a menos que esse endereço IP seja reutilizado por um novo nó no cluster.

Para backends preexistentes, atualizar o backend com `tridentctl update backend` garante que Trident gerencie as políticas de exportação automaticamente. Isso cria duas novas políticas de exportação nomeadas de acordo com o UUID do backend e o nome da qtree quando necessário. Volumes presentes no backend usarão as novas políticas de exportação após serem desmontados e montados novamente.

> A exclusão de um backend com políticas de exportação gerenciadas automaticamente excluirá a política de exportação criada dinamicamente. Se o backend for recriado, ele será tratado como um novo backend e resultará na criação de uma nova política de exportação.

Se o endereço IP de um nó ativo for atualizado, você deve reiniciar o pod do Trident nesse nó. Trident então atualizará a política de exportação dos backends que gerencia para refletir essa alteração de IP.

Prepare-se para provisionar volumes SMB

Com um pouco de preparação adicional, você pode provisionar volumes SMB usando `ontap-nas` drivers.

> Você deve configurar ambos os protocolos NFS e SMB/CIFS na SVM para criar um `ontap-nas-economy` volume SMB para clusters ONTAP locais. A falha ao configurar qualquer um desses protocolos fará com que a criação do volume SMB falhe.

> `autoExportPolicy` não é compatível com volumes SMB.

Antes de começar

Antes de poder provisionar volumes SMB, você deve ter o seguinte.

- Um cluster Kubernetes com um nó controlador Linux e pelo menos um nó de trabalho Windows executando Windows Server 2022. Trident suporta volumes SMB montados em pods executados apenas em nós Windows.
- Pelo menos um segredo Trident contendo suas credenciais do Active Directory. Para gerar o segredo `smbcreds`:

```
kubectl create secret generic smbcreds --from-literal username=user
--from-literal password='password'
```

- Um proxy CSI configurado como um serviço do Windows. Para configurar um `csi-proxy`, consulte "GitHub: CSI Proxy" ou "GitHub: CSI Proxy para Windows" para nós do Kubernetes em execução no Windows.

Passos

1. Para o ONTAP local, você pode opcionalmente criar um compartilhamento SMB ou o Trident pode criar um para você.

   Os compartilhamentos SMB são necessários para Amazon FSx for ONTAP.

   Você pode criar os compartilhamentos administrativos SMB de duas maneiras: usando o "Microsoft Management Console" snap-in Shared Folders ou usando a ONTAP CLI. Para criar os compartilhamentos SMB usando a ONTAP CLI:

   a. Se necessário, crie a estrutura de caminho de diretórios para o compartilhamento.

      O `vserver cifs share create` comando verifica o caminho especificado na opção -path durante a criação do compartilhamento. Se o caminho especificado não existir, o comando falha.

   b. Crie um compartilhamento SMB associado à SVM especificada:

```
vserver cifs share create -vserver vserver_name -share-name
share_name -path path [-share-properties share_properties,...]
[other_attributes] [-comment text]
```

   c. Verifique se o compartilhamento foi criado:

```
vserver cifs share show -share-name share_name
```

   Consulte "Criar um compartilhamento SMB" para obter detalhes completos.

2. Ao criar o backend, você deve configurar o seguinte para especificar os volumes SMB. Para todas as opções de configuração do backend do FSx para ONTAP, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Parâmetro	Descrição	Exemplo
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`nasType`	**Deve ser definido como `smb`.** Se for nulo, o padrão é `nfs`.	`smb`
`securityStyle`	Estilo de segurança para novos volumes. **Deve ser definido como `ntfs` ou `mixed` para volumes SMB.**	`ntfs` ou `mixed` for SMB volumes
`unixPermissions`	Modo para novos volumes. **Deve ser deixado em branco para volumes SMB.**	""

Ativar SMB seguro

A partir da versão 25.06, NetApp Trident oferece suporte ao provisionamento seguro de volumes SMB criados usando `ontap-nas` e `ontap-nas-economy` backends. Quando o SMB seguro está habilitado, você pode fornecer acesso controlado aos compartilhamentos SMB para usuários e grupos de usuários do Active Directory (AD) usando listas de controle de acesso (ACLs).

Pontos a lembrar

- A importação de `ontap-nas-economy` volumes não é suportada.
- Apenas clones somente leitura são suportados para `ontap-nas-economy` volumes.
- Se o Secure SMB estiver ativado, Trident ignorará o compartilhamento SMB mencionado no backend.
- A atualização da anotação PVC, da anotação da storage class e do campo backend não atualiza a ACL de compartilhamento SMB.
- A ACL de compartilhamento SMB especificada na anotação do PVC clonado terá precedência sobre as do PVC de origem.
- Certifique-se de fornecer usuários válidos do Active Directory ao habilitar o SMB seguro. Usuários inválidos não serão adicionados à ACL.
- Se você fornecer o mesmo usuário do AD no backend, na storage class e no PVC com permissões diferentes, a prioridade de permissão será: PVC, storage class e, em seguida, backend.
- Secure SMB é compatível com `ontap-nas`importações de volumes gerenciados e não se aplica a importações de volumes não gerenciados.

Passos

1. Especifique adAdminUser em TridentBackendConfig conforme mostrado no exemplo a seguir:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.193.176.x
  svm: svm0
  useREST: true
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

2. Adicione a anotação na storage class.

   Adicione a `trident.netapp.io/smbShareAdUser` anotação à storage class para habilitar SMB seguro sem falhas. O valor de usuário especificado para a anotação `trident.netapp.io/smbShareAdUser` deve ser o mesmo que o nome de usuário especificado no `smbcreds` secret. Você pode escolher um dos seguintes para `smbShareAdUserPermission`: `full_control`, `change` ou `read`. A permissão padrão é `full_control`.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

1. Crie um PVC.

   O exemplo a seguir cria um PVC:

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/snapshotDirectory: "true"
    trident.netapp.io/smbShareAccessControl: |
      read:
        - tridentADtest
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Opções e exemplos de configuração do ONTAP NAS

Aprenda a criar e usar drivers ONTAP NAS com sua instalação do Trident. Esta seção fornece exemplos de configuração de backend e detalhes para mapear backends para StorageClasses.

A partir da versão 25.10, NetApp Trident oferece suporte "NetApp sistemas de storage AFX". NetApp AFX sistemas de storage diferem de outros sistemas baseados em ONTAP (ASA, AFF e FAS) na implementação de sua camada de storage.

> Apenas o `ontap-nas`driver (com protocolo NFS) é compatível com sistemas AFX da NetApp; o protocolo SMB não é compatível.

Na configuração do backend do Trident, não é necessário especificar que seu sistema é um NetApp AFX sistema de storage. Ao selecionar `ontap-nas` como o `storageDriverName`, o Trident detecta automaticamente o sistema de storage AFX. Alguns parâmetros de configuração do backend não se aplicam a sistemas de storage AFX, conforme indicado na tabela abaixo.

Opções de configuração do backend

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Padrão
`version`		Sempre 1

Parâmetro	Descrição	Padrão
`storageDriverName`	Nome do driver de armazenamento Para sistemas NetApp AFX, apenas `ontap-nas` é suportado.	`ontap-nas`, `ontap-nas-economy`, ou `ontap-nas-flexgroup`
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. Para um switchover MetroCluster contínuo, consulte o Exemplo do MetroCluster.	"10.0.0.1", "[2001:1234:abcd::fefe]"
`dataLIF`	Endereço IP do protocolo LIF. NetApp recomenda especificar `dataLIF`. Se não for fornecido, Trident busca os dataLIFs da SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Pode ser alterado após a configuração inicial. Consulte . Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como `[28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]`. **Omitir para MetroCluster.** Consulte Exemplo do MetroCluster.	Endereço especificado ou derivado de SVM, caso não seja especificado (não recomendado)
`svm`	Máquina virtual de storage a ser usada **Omitir para MetroCluster.** Consulte o Exemplo do MetroCluster.	Derivado se uma SVM `managementLIF` for especificada
`autoExportPolicy`	Ativar a criação e atualização automática de políticas de exportação [Booleano]. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	falso
`autoExportCIDRs`	Lista de CIDRs para filtrar os IPs dos nós do Kubernetes quando `autoExportPolicy` estiver habilitado. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	["0.0.0.0/0", "::/0"]`
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`clientCertificate`	Valor codificado em Base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Padrão
`clientPrivateKey`	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em Base64 do certificado da Autoridade Certificadora (CA) confiável. Opcional. Usado para autenticação baseada em certificado	""
`username`	Nome de usuário para conectar-se ao cluster/SVM. Usado para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	
`password`	Senha para conectar-se ao cluster/SVM. Usada para autenticação baseada em credenciais. Para autenticação do Active Directory, consulte "Autentique o Trident em um SVM de backend usando credenciais do Active Directory".	
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser atualizado após ser definido Ao usar o ontap-nas-economy e um storagePrefix com 24 ou mais caracteres, as qtrees não terão o storage prefix incorporado, embora ele esteja presente no nome do volume.	"Trident"

Parâmetro	Descrição	Padrão
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Se não for atribuído, qualquer um dos agregados disponíveis pode ser usado para provisionar um FlexGroup volume. Quando o agregado é atualizado no SVM, ele é atualizado automaticamente no Trident por meio de polling no SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou um agregado específico no Trident para provisionar volumes, se o agregado for renomeado ou movido para fora do SVM, o backend entrará em estado de falha no Trident durante o polling do agregado no SVM. Você deve alterar o agregado para um que esteja presente no SVM ou removê-lo completamente para que o backend volte a ficar online. **Não especifique para sistemas de armazenamento AFX.**	""
`limitAggregateUsage`	O provisionamento falhará se o uso ultrapassar essa porcentagem. **Não se aplica ao Amazon FSx para ONTAP**. **Não especifique para sistemas de storage AFX.**	"" (não aplicado por padrão)

Parâmetro	Descrição	Padrão
flexgroupAggregateList	Lista de agregados para provisionamento (opcional; se definida, deve ser atribuída à SVM). Todos os agregados atribuídos à SVM são usados para provisionar um FlexGroup volume. Compatível com o driver de storage **ontap-nas-flexgroup**. Quando a lista de agregados é atualizada no SVM, a lista é atualizada automaticamente no Trident por meio de polling do SVM, sem a necessidade de reiniciar o Trident Controller. Quando você configurou uma lista de agregados específica no Trident para provisionar volumes, se a lista de agregados for renomeada ou movida para fora do SVM, o backend entrará em estado de falha no Trident enquanto faz polling do agregado do SVM. Você deve alterar a lista de agregados para uma que esteja presente no SVM ou removê-la completamente para que o backend volte a ficar online.	""
`limitVolumeSize`	Falhe no provisionamento se o tamanho do volume solicitado for superior a este valor.	"" (não aplicado por padrão)
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} não use `debugTraceFlags` a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nasType`	Configure a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. Definir como null utiliza volumes NFS por padrão. **Se especificado, defina sempre como `nfs` para sistemas de storage AFX**.	`nfs`
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes são normalmente especificadas nas classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, Trident usará as opções de montagem especificadas no arquivo de configuração do backend de storage. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`qtreesPerFlexvol`	Máximo de Qtrees por FlexVol, deve estar no intervalo [50, 300]	"200"

Parâmetro	Descrição	Padrão
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI; um nome para permitir que o Trident crie o compartilhamento SMB; ou você pode deixar o parâmetro em branco para impedir o acesso comum aos volumes. Este parâmetro é opcional para ONTAP on-premises. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends e não pode ficar em branco.	`smb-share`
`useREST`	Parâmetro booleano para usar as APIs REST do ONTAP. `useREST`Quando definido como `true`, Trident usa as APIs REST do ONTAP para se comunicar com o backend; quando definido como `false`, Trident usa chamadas ONTAPI (ZAPI) para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 ou posterior. Além disso, a função de login do ONTAP usada deve ter acesso ao aplicativo `ontapi`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`. A partir da versão 24.06 do Trident e ONTAP 9.15.1 ou posterior, `useREST` é definido como `true` por padrão; altere `useREST` para `false` para usar chamadas ONTAPI (ZAPI). **Se especificado, defina sempre como `true` para sistemas de armazenamento AFX**.	`true`para ONTAP 9.15.1 ou posterior, caso contrário `false`.
`limitVolumePoolSize`	Tamanho máximo solicitável de FlexVol ao usar Qtrees no backend ontap-nas-economy.	"" (não aplicado por padrão)
`denyNewVolumePools`	Restringe `ontap-nas-economy` os backends de criarem novos volumes FlexVol para conter seus Qtrees. Somente FlexVols preexistentes são usados para provisionar novos PVs.	
`adAdminUser`	Usuário ou grupo de usuários administradores do Active Directory com acesso total aos compartilhamentos SMB. Use este parâmetro para conceder direitos de administrador ao compartilhamento SMB com controle total.	

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para Qtrees	"true"
`spaceReserve`	Modo de reserva de espaço; "none" (fino) ou "volume" (grosso)	"none"

Parâmetro	Descrição	Padrão
`snapshotPolicy`	Política do Snapshot a ser usada	"none"
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha um de qosPolicy ou adaptiveQosPolicy por pool de storage/backend	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável a ser atribuído aos volumes criados. Escolha uma das opções qosPolicy ou adaptiveQosPolicy por pool de storage/backend. Não compatível com ontap-nas-economy.	""
`snapshotReserve`	Percentual do volume reservado para snapshots	"0" se `snapshotPolicy` for "none", caso contrário ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	"false"
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	"false"
`tieringPolicy`	Política de tiering para usar "none"	
`unixPermissions`	Modo para novos volumes	"777" para volumes NFS; vazio (não aplicável) para volumes SMB
`snapshotDir`	Controla o acesso ao `.snapshot` diretório	"true" para NFSv4 "false" para NFSv3
`exportPolicy`	Política de exportação a ser usada	"default"
`securityStyle`	Estilo de segurança para novos volumes. NFS suporta `mixed` e `unix` estilos de segurança. SMB suporta `mixed` e `ntfs` estilos de segurança.	NFS default é `unix`. SMB default é `ntfs`.
`nameTemplate`	Modelo para criar nomes de volume personalizados.	""

Usar grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o teto para a taxa de transferência total de todas as cargas de trabalho.

Exemplos de provisionamento de volume

Aqui está um exemplo com valores padrão definidos:

```
---
version: 1
storageDriverName: ontap-nas
backendName: customBackendName
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
labels:
  k8scluster: dev1
  backend: dev1-nasbackend
svm: trident_svm
username: cluster-admin
password: <password>
limitAggregateUsage: 80%
limitVolumeSize: 50Gi
nfsMountOptions: nfsvers=4
debugTraceFlags:
  api: false
  method: true
defaults:
  spaceReserve: volume
  qosPolicy: premium
  exportPolicy: myk8scluster
  snapshotPolicy: default
  snapshotReserve: "10"
```

Para `ontap-nas` e `ontap-nas-flexgroups`, o Trident agora usa um novo cálculo para garantir que o FlexVol seja dimensionado corretamente com a porcentagem de snapshotReserve e o PVC. Quando o usuário solicita um PVC, o Trident cria o FlexVol original com mais espaço usando o novo cálculo. Esse cálculo garante que o usuário receba o espaço gravável solicitado no PVC, e não menos espaço do que o solicitado. Antes da v21.07, quando o usuário solicitava um PVC (por exemplo, 5 GiB), com o snapshotReserve em 50 por cento, ele recebia apenas 2,5 GiB de espaço gravável. Isso ocorre porque o que o usuário solicitava era o volume total e `snapshotReserve` é uma porcentagem disso. Com o Trident 21.07, o que o usuário solicita é o espaço gravável e o Trident define o número de `snapshotReserve` como a porcentagem do volume total. Isso não se aplica a `ontap-nas-economy`. Veja o exemplo a seguir para ver como isso funciona:

O cálculo é o seguinte:

```
Total volume size = <PVC requested size> / (1 - (<snapshotReserve
percentage> / 100))
```

Para snapshotReserve = 50%, e solicitação de PVC = 5 GiB, o tamanho total do volume é 5/0,5 = 10 GiB e o tamanho disponível é 5 GiB, que é o que o usuário solicitou na solicitação de PVC. O `volume show` comando deve mostrar resultados semelhantes a este exemplo:

Os backends existentes de instalações anteriores provisionarão volumes conforme explicado acima ao atualizar Trident. Para volumes que você criou antes da atualização, você deve redimensionar seus volumes para que a alteração seja observada. Por exemplo, um PVC de 2 GiB com `snapshotReserve=50` anteriormente resultava em um volume que fornece 1 GiB de espaço gravável. Redimensionar o volume para 3 GiB, por exemplo, fornece ao aplicativo 3 GiB de espaço gravável em um volume de 6 GiB.

Exemplos de configuração mínima

Os exemplos a seguir mostram configurações básicas que deixam a maioria dos parâmetros com os valores padrão. Esta é a maneira mais fácil de definir um backend.

> Se você estiver usando Amazon FSx no NetApp ONTAP com Trident, a recomendação é especificar nomes DNS para LIFs em vez de endereços IP.

Exemplo de ONTAP NAS economy

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo de FlexGroup ONTAP NAS

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo do MetroCluster

Você pode configurar o backend para evitar ter que atualizar manualmente a definição do backend após switchover e switchback durante "Replicação e recuperação de SVM".

Para switchover e switchback sem interrupções, especifique a SVM usando `managementLIF` e omita os parâmetros `dataLIF` e `svm`. Por exemplo:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 192.168.1.66
username: vsadmin
password: password
```

Exemplo de volumes SMB

```
---
version: 1
backendName: ExampleBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
nasType: smb
securityStyle: ntfs
unixPermissions: ""
dataLIF: 10.0.0.2
svm: svm_nfs
username: vsadmin
password: password
```

Exemplo de autenticação baseada em certificado

Este é um exemplo mínimo de configuração de backend. `clientCertificate`, `clientPrivateKey` e `trustedCACertificate` (opcional, se estiver usando uma CA confiável) são preenchidos em `backend.json` e recebem os valores codificados em base64 do certificado do cliente, da chave privada e do certificado da CA confiável, respectivamente.

```
---
version: 1
backendName: DefaultNASBackend
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.15
svm: nfs_svm
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemplo de política de exportação automática

Este exemplo mostra como você pode instruir Trident a usar políticas de exportação dinâmicas para criar e gerenciar a política de exportação automaticamente. Isso funciona da mesma forma para os `ontap-nas-economy` e `ontap-nas-flexgroup`drivers.

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: svm_nfs
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-nasbackend
autoExportPolicy: true
autoExportCIDRs:
- 10.0.0.0/24
username: admin
password: password
nfsMountOptions: nfsvers=4
```

Exemplo de endereços IPv6

Este exemplo mostra `managementLIF` usando um endereço IPv6.

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas_ipv6_backend
managementLIF: "[5c5d:5edf:8f:7657:bef8:109b:1b41:d491]"
labels:
  k8scluster: test-cluster-east-1a
  backend: test1-ontap-ipv6
svm: nas_ipv6_svm
username: vsadmin
password: password
```

Amazon FSx para ONTAP usando exemplo de volumes SMB

O `smbShare` parâmetro é necessário para Amazon FSx para ONTAP usando volumes SMB.

```
---
version: 1
backendName: SMBBackend
storageDriverName: ontap-nas
managementLIF: example.mgmt.fqdn.aws.com
nasType: smb
dataLIF: 10.0.0.15
svm: nfs_svm
smbShare: smb-share
clientCertificate: ZXR0ZXJwYXB...ICMgJ3BhcGVyc2
clientPrivateKey: vciwKIyAgZG...0cnksIGRlc2NyaX
trustedCACertificate: zcyBbaG...b3Igb3duIGNsYXNz
storagePrefix: myPrefix_
```

Exemplo de configuração de backend com nameTemplate

```
---
version: 1
storageDriverName: ontap-nas
backendName: ontap-nas-backend
managementLIF: <ip address>
svm: svm0
username: <admin>
password: <password>
defaults:
  nameTemplate:
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.vo\
    lume.RequestName}}"
labels:
  cluster: ClusterA
  PVC: "{{.volume.Namespace}}_{{.volume.RequestName}}"
```

Exemplos de backends com pools virtuais

Nos arquivos de definição de backend de exemplo mostrados abaixo, valores padrão específicos são definidos para todos os pools de storage, como `spaceReserve` em none, `spaceAllocation` em false e `encryption` em false. Os pools virtuais são definidos na seção de storage.

Trident define os rótulos de provisionamento no campo "Comentários". Os comentários são definidos em FlexVol para `ontap-nas` ou FlexGroup para `ontap-nas-flexgroup`. Trident copia todos os rótulos presentes em um pool virtual para o volume de armazenamento durante o provisionamento. Para conveniência, os administradores de armazenamento podem definir rótulos por pool virtual e agrupar volumes por rótulo.

Nestes exemplos, alguns pools de storage definem seus próprios `spaceReserve`, `spaceAllocation`, e `encryption` valores, e alguns pools substituem os valores padrão.

Exemplo de ONTAP NAS

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 10.0.0.1
svm: svm_nfs
username: admin
password: <password>
nfsMountOptions: nfsvers=4
defaults:
  spaceReserve: none
  encryption: "false"
  qosPolicy: standard
labels:
  store: nas_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      app: msoffice
      cost: "100"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
      adaptiveQosPolicy: adaptive-premium
  - labels:
      app: slack
      cost: "75"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: legal
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      app: wordpress
```

```
      cost: "50"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      app: mysqldb
      cost: "25"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemplo de ONTAP NAS FlexGroup

```
---
version: 1
storageDriverName: ontap-nas-flexgroup
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: flexgroup_store
  k8scluster: prod-cluster-1
region: us_east_1
storage:
  - labels:
      protection: gold
      creditpoints: "50000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: gold
      creditpoints: "30000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: silver
      creditpoints: "20000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      protection: bronze
      creditpoints: "10000"
    zone: us_east_1d
    defaults:
```

```
      spaceReserve: volume
      encryption: "false"
      unixPermissions: "0775"
```

Exemplo de ONTAP NAS economy

```
---
version: 1
storageDriverName: ontap-nas-economy
managementLIF: 10.0.0.1
svm: svm_nfs
username: vsadmin
password: <password>
defaults:
  spaceReserve: none
  encryption: "false"
labels:
  store: nas_economy_store
region: us_east_1
storage:
  - labels:
      department: finance
      creditpoints: "6000"
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      protection: bronze
      creditpoints: "5000"
    zone: us_east_1b
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0755"
  - labels:
      department: engineering
      creditpoints: "3000"
    zone: us_east_1c
    defaults:
      spaceReserve: none
      encryption: "true"
      unixPermissions: "0775"
  - labels:
      department: humanresource
      creditpoints: "2000"
    zone: us_east_1d
    defaults:
      spaceReserve: volume
```

```
      encryption: "false"
      unixPermissions: "0775"
```

Mapear back-ends para StorageClasses

As seguintes definições de StorageClass referem-se a Exemplos de backends com pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume. O volume terá os aspectos definidos no pool virtual escolhido.

- O `protection-gold` StorageClass corresponderá ao primeiro e ao segundo pool virtual no `ontap-nas-flexgroup` backend. Esses são os únicos pools que oferecem proteção de nível ouro.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=gold"
  fsType: "ext4"
```

- O `protection-not-gold` StorageClass corresponderá ao terceiro e quarto pool virtual no `ontap-nas-flexgroup` backend. Esses são os únicos pools que oferecem nível de proteção diferente de gold.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-not-gold
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection!=gold"
  fsType: "ext4"
```

- O `app-mysqldb` StorageClass será mapeado para o quarto pool virtual no `ontap-nas` backend. Este é o único pool que oferece configuração de pool de storage para app do tipo mysqldb.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: app-mysqldb
provisioner: csi.trident.netapp.io
parameters:
  selector: "app=mysqldb"
  fsType: "ext4"
```

- O `protection-silver-creditpoints-20k` StorageClass será mapeado para o terceiro pool virtual no `ontap-nas-flexgroup` backend. Este é o único pool que oferece proteção de nível prata e 20000 pontos de crédito.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: protection-silver-creditpoints-20k
provisioner: csi.trident.netapp.io
parameters:
  selector: "protection=silver; creditpoints=20000"
  fsType: "ext4"
```

- O `creditpoints-5k` StorageClass corresponderá ao terceiro pool virtual no `ontap-nas` backend e ao segundo pool virtual no `ontap-nas-economy` backend. Essas são as únicas ofertas de pool com 5000 creditpoints.

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: creditpoints-5k
provisioner: csi.trident.netapp.io
parameters:
  selector: "creditpoints=5000"
  fsType: "ext4"
```

Trident decidirá qual pool virtual será selecionado e garantirá que o requisito de storage seja atendido.

Atualize `dataLIF` após a configuração inicial

Você pode alterar o dataLIF após a configuração inicial executando o seguinte comando para fornecer o novo arquivo JSON de backend com o dataLIF atualizado.

```
tridentctl update backend <backend-name> -f <path-to-backend-json-file-
with-updated-dataLIF>
```

Se os PVCs estiverem conectados a um ou mais pods, você deve desligar todos os pods correspondentes e então ligá-los novamente para que o novo dataLIF entre em vigor.

Exemplos seguros de SMB

Configuração de backend com o driver ontap-nas

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com o driver ontap-nas-economy

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas-economy
  managementLIF: 10.0.0.1
  svm: svm2
  nasType: smb
  defaults:
    adAdminUser: tridentADtest
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Configuração de backend com pool de storage

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.0.0.1
  svm: svm0
  useREST: false
  storage:
  - labels:
      app: msoffice
    defaults:
      adAdminUser: tridentADuser
  nasType: smb
  credentials:
    name: backend-tbc-ontap-invest-secret
```

Exemplo de classe de armazenamento com o driver ontap-nas

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADtest
parameters:
  backendType: ontap-nas
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Certifique-se de adicionar `annotations` para habilitar o SMB seguro. O SMB seguro não funciona sem as anotações, independentemente das configurações definidas no Backend ou no PVC.

Exemplo de classe de armazenamento com o driver ontap-nas-economy

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-smb-sc
  annotations:
    trident.netapp.io/smbShareAdUserPermission: change
    trident.netapp.io/smbShareAdUser: tridentADuser3
parameters:
  backendType: ontap-nas-economy
  csi.storage.k8s.io/node-stage-secret-name: smbcreds
  csi.storage.k8s.io/node-stage-secret-namespace: trident
  trident.netapp.io/nasType: smb
provisioner: csi.trident.netapp.io
reclaimPolicy: Delete
volumeBindingMode: Immediate
```

Exemplo de PVC com um único usuário AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-pvc4
  namespace: trident
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      change:
        - tridentADtest
      read:
        - tridentADuser
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-smb-sc
```

Exemplo de PVC com vários usuários de AD

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: my-test-pvc
  annotations:
    trident.netapp.io/smbShareAccessControl: |
      full_control:
        - tridentTestuser
        - tridentuser
        - tridentTestuser1
        - tridentuser1
      change:
        - tridentADuser
        - tridentADuser1
        - tridentADuser4
        - tridentTestuser2
      read:
        - tridentTestuser2
        - tridentTestuser3
        - tridentADuser2
        - tridentADuser3
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
```

Amazon FSx for NetApp ONTAP

Use Trident com Amazon FSx for NetApp ONTAP

"Amazon FSx for NetApp ONTAP" é um serviço totalmente gerenciado da AWS que permite aos clientes lançar e executar sistemas de arquivos baseados no sistema operacional de storage NetApp ONTAP. FSx for ONTAP permite que você aproveite os recursos, o desempenho e as capacidades administrativas do NetApp com os quais você já está familiarizado, enquanto aproveita a simplicidade, agilidade, segurança e escalabilidade de armazenar dados na AWS. FSx for ONTAP oferece suporte aos recursos do sistema de arquivos ONTAP e às APIs de administração.

Você pode integrar seu Amazon FSx for NetApp ONTAP file system com Trident para garantir que clusters Kubernetes em execução no Amazon Elastic Kubernetes Service (EKS) possam provisionar volumes persistentes de bloco e de arquivo com suporte do ONTAP.

Um sistema de arquivos é o recurso principal no Amazon FSx, análogo a um cluster ONTAP em infraestrutura local. Dentro de cada SVM você pode criar um ou mais volumes, que são contêineres de dados que

armazenam os arquivos e pastas no seu sistema de arquivos. Com Amazon FSx for NetApp ONTAP será fornecido como um sistema de arquivos gerenciado na nuvem. O novo tipo de sistema de arquivos é chamado **NetApp ONTAP**.

Ao usar Trident com Amazon FSx para NetApp ONTAP, você pode garantir que clusters Kubernetes em execução no Amazon Elastic Kubernetes Service (EKS) possam provisionar volumes persistentes em bloco e em arquivo com suporte do ONTAP.

Requisitos

Além de "Requisitos do Trident", para integrar o FSx for ONTAP com Trident, você precisa de:

- Um cluster Amazon EKS existente ou um cluster Kubernetes autogerenciado com `kubectl` instalado.
- Um sistema de arquivos Amazon FSx for NetApp ONTAP e uma máquina virtual de armazenamento (SVM) existentes que sejam acessíveis a partir dos nós de trabalho do seu cluster.
- Nós de trabalho preparados para "NFS ou iSCSI".

> Certifique-se de seguir os passos de preparação do nó necessários para Amazon Linux e Ubuntu "Amazon Machine Images" (AMIs), dependendo do seu tipo de AMI do EKS.

Considerações

- Volumes SMB:
  - Os volumes SMB são suportados usando apenas o driver `ontap-nas`.
  - Os volumes SMB não são suportados com o Trident EKS add-on.
  - Trident suporta volumes SMB montados em pods executados apenas em nós Windows. Consulte "Prepare-se para provisionar volumes SMB" para obter detalhes.
- Antes do Trident 24.02, volumes criados em sistemas de arquivos Amazon FSx que tinham backups automáticos ativados não podiam ser excluídos pelo Trident. Para evitar esse problema no Trident 24.02 ou posterior, especifique o `fsxFilesystemID`, AWS `apiRegion`, AWS `apikey` e AWS `secretKey` no arquivo de configuração do backend para AWS FSx for ONTAP.

> Se você estiver especificando uma função do IAM para Trident, poderá omitir a especificação dos campos `apiRegion`, `apiKey` e `secretKey` para o Trident explicitamente. Para mais informações, consulte "Opções e exemplos de configuração do FSx for ONTAP".

Uso simultâneo do Trident SAN/iSCSI e do driver EBS-CSI

Se você planeja usar drivers ontap-san (por exemplo, iSCSI) com AWS (EKS, ROSA, EC2 ou qualquer outra instância), a configuração de multipath necessária nos nós pode entrar em conflito com o driver CSI do Amazon Elastic Block Store (EBS). Para garantir que o multipath funcione sem interferir nos discos EBS no mesmo nó, você precisa excluir EBS da sua configuração de multipath. Este exemplo mostra um `multipath.conf`arquivo que inclui as configurações necessárias do Trident, excluindo os discos EBS do multipath:

```
defaults {
    find_multipaths no
}
blacklist {
    device {
        vendor "NVME"
        product "Amazon Elastic Block Store"
    }
}
```

Autenticação

Trident oferece dois modos de autenticação.

- Baseado em credenciais (recomendado): armazena as credenciais com segurança no AWS Secrets Manager. Você pode usar o `fsxadmin` usuário do seu sistema de arquivos ou o `vsadmin` usuário configurado para sua SVM.

> Trident espera ser executado como um `vsadmin` usuário SVM ou como um usuário com um nome diferente que tenha a mesma função. Amazon FSx for NetApp ONTAP possui um `fsxadmin` usuário que é um substituto limitado para o ONTAP `admin` cluster user. Recomendamos fortemente usar `vsadmin` com Trident.

- Com base em certificado: Trident se comunicará com a SVM no seu sistema de arquivos FSx usando um certificado instalado na sua SVM.

Para obter detalhes sobre como ativar a autenticação, consulte a autenticação para o seu tipo de driver:

- "ONTAP NAS autenticação"
- "ONTAP SAN autenticação"

Imagens de Máquina da Amazon (AMIs) testadas

O cluster EKS suporta diversos sistemas operacionais, mas a AWS otimizou determinadas Amazon Machine Images (AMIs) para contêineres e EKS. As seguintes AMIs foram testadas com NetApp Trident 25.02.

AMI	NAS	NAS-economy	iSCSI	iSCSI-economia
AL2023_x86_64_STANDARD	Sim	Sim	Sim	Sim
AL2_x86_64	Sim	Sim	Sim*	Sim*
BOTTLEROCKET_x86_64	Sim**	Sim	N/A	N/A
AL2023_ARM_64_STANDARD	Sim	Sim	Sim	Sim
AL2_ARM_64	Sim	Sim	Sim*	Sim*

BOTTLEROCKET_ARM_64	Sim**	Sim	N/A	N/A

- * Não foi possível excluir o PV sem reiniciar o nó
- ** Não funciona com NFSv3 com Trident versão 25.02.

> Se a AMI desejada não estiver listada aqui, isso não significa que ela não seja compatível; significa apenas que ela não foi testada. Esta lista serve como um guia para AMIs que comprovadamente funcionam.

Testes realizados com:

- Versão do EKS: 1.32
- Método de instalação: Helm 25.06 e como um AWS add-On 25.06
- Para NAS, foram testados tanto NFSv3 quanto NFSv4.1.
- Para SAN, apenas iSCSI foi testado, não NVMe-oF.

Testes realizados:

- Criar: classe de armazenamento, pvc, pod
- Excluir: pod, pvc (regular, qtree/lun – econômico, NAS com backup da AWS)

Encontre mais informações

- "Documentação do Amazon FSx for NetApp ONTAP"
- "Postagem no blog sobre Amazon FSx for NetApp ONTAP"

Crie uma função do IAM e um segredo da AWS

Você pode configurar pods do Kubernetes para acessar recursos da AWS autenticando-se como uma função do AWS IAM em vez de fornecer credenciais explícitas da AWS.

> Para autenticar usando uma função do AWS IAM, você deve ter um cluster Kubernetes implantado usando EKS.

Criar segredo do AWS Secrets Manager

Como Trident emitirá APIs contra um FSx vserver para gerenciar o storage para você, ele precisará de credenciais para isso. A maneira segura de passar essas credenciais é por meio de um segredo do AWS Secrets Manager. Portanto, se você ainda não tiver um, precisará criar um segredo do AWS Secrets Manager que contenha as credenciais da conta vsadmin.

Este exemplo cria um segredo do AWS Secrets Manager para armazenar credenciais do Trident CSI:

```
aws secretsmanager create-secret --name trident-secret --description
"Trident CSI credentials"\
  --secret-string
"{\"username\":\"vsadmin\",\"password\":\"<svmpassword>\"}"
```

Criar política de IAM

Trident também precisa de permissões da AWS para funcionar corretamente. Portanto, você precisa criar uma política que conceda ao Trident as permissões de que ele precisa.

Os exemplos a seguir criam uma política do IAM usando a AWS CLI:

```
aws iam create-policy --policy-name AmazonFSxNCSIDriverPolicy --policy
-document file://policy.json
  --description "This policy grants access to Trident CSI to FSxN and
Secrets manager"
```

Exemplo de JSON de policy:

```
{
  "Statement": [
    {
      "Action": [
        "fsx:DescribeFileSystems",
        "fsx:DescribeVolumes",
        "fsx:CreateVolume",
        "fsx:RestoreVolumeFromSnapshot",
        "fsx:DescribeStorageVirtualMachines",
        "fsx:UntagResource",
        "fsx:UpdateVolume",
        "fsx:TagResource",
        "fsx:DeleteVolume"
      ],
      "Effect": "Allow",
      "Resource": "*"
    },
    {
      "Action": "secretsmanager:GetSecretValue",
      "Effect": "Allow",
      "Resource": "arn:aws:secretsmanager:<aws-region>:<aws-account-
id>:secret:<aws-secret-manager-name>*"
    }
  ],
  "Version": "2012-10-17"
}
```

Criar identidade de Pod ou função IAM para associação de conta de serviço (IRSA)

Você pode configurar uma conta de serviço do Kubernetes para assumir uma função do AWS Identity and Access Management (IAM) com EKS Pod Identity ou IAM role for Service account association (IRSA). Quaisquer Pods configurados para usar a conta de serviço podem então acessar qualquer serviço da AWS ao

qual a função tenha permissões de acesso.

Identidade do Pod

As associações de identidade de pods do Amazon EKS permitem gerenciar credenciais para seus aplicativos, de forma semelhante à maneira como os perfis de instância do Amazon EC2 fornecem credenciais para instâncias do Amazon EC2.

Instale o Pod Identity no seu cluster EKS:

Você pode criar uma identidade de Pod via console da AWS ou usando o seguinte comando da AWS CLI:

```
aws eks create-addon --cluster-name <EKS_CLUSTER_NAME> --addon-name
eks-pod-identity-agent
```

Para obter mais informações, consulte "Configure o agente de identidade do Amazon EKS Pod".

Crie trust-relationship.json:

Crie o arquivo trust-relationship.json para permitir que o EKS Service Principal assuma essa função para Pod Identity. Em seguida, crie uma função com esta trust policy:

```
aws iam create-role \
  --role-name fsxn-csi-role --assume-role-policy-document file://trust-
relationship.json \
  --description "fsxn csi pod identity role"
```

arquivo trust-relationship.json:

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Service": "pods.eks.amazonaws.com"
      },
      "Action": [
        "sts:AssumeRole",
        "sts:TagSession"
      ]
    }
  ]
}
```

Anexe a política de função à função do IAM:

Anexe a política de função da etapa anterior à função IAM que foi criada:

```
aws iam attach-role-policy \
  --policy-arn arn:aws:iam::aws:111122223333:policy/fsxn-csi-policy \
  --role-name fsxn-csi-role
```

Crie uma associação de identidade de pod:

Crie uma associação de identidade de pod entre a função IAM e a conta de serviço Trident (trident-controller)

```
aws eks create-pod-identity-association \
  --cluster-name <EKS_CLUSTER_NAME> \
  --role-arn arn:aws:iam::111122223333:role/fsxn-csi-role \
  --namespace trident --service-account trident-controller
```

Função IAM para associação de conta de serviço (IRSA)

Usando a AWS CLI:

```
aws iam create-role --role-name AmazonEKS_FSxN_CSI_DriverRole \
  --assume-role-policy-document file://trust-relationship.json
```

Arquivo trust-relationship.json

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Principal": {
        "Federated": "arn:aws:iam::<account_id>:oidc-
provider/<oidc_provider>"
      },
      "Action": "sts:AssumeRoleWithWebIdentity",
      "Condition": {
        "StringEquals": {
          "<oidc_provider>:aud": "sts.amazonaws.com",
          "<oidc_provider>:sub":
"system:serviceaccount:trident:trident-controller"
        }
      }
    }
  ]
}
```

Atualize os seguintes valores no arquivo `trust-relationship.json`:

- **<account_id>** - Seu ID de conta da AWS
- **<oidc_provider>** - O OIDC do seu cluster EKS. Você pode obter o oidc_provider executando:

```
aws eks describe-cluster --name my-cluster --query
"cluster.identity.oidc.issuer"\
  --output text | sed -e "s/^https:\/\///"
```

Associe a função IAM à política IAM:

Depois que a função for criada, associe a política (que foi criada na etapa acima) à função usando este comando:

```
aws iam attach-role-policy --role-name my-role --policy-arn <IAM policy
ARN>
```

Verifique se o provedor do OICD está associado:

Verifique se o seu provedor OIDC está associado ao seu cluster. Você pode verificar isso usando este comando:

```
aws iam list-open-id-connect-providers | grep $oidc_id | cut -d "/" -f4
```

Se a saída estiver vazia, use o seguinte comando para associar IAM OIDC ao seu cluster:

```
eksctl utils associate-iam-oidc-provider --cluster $cluster_name
--approve
```

Se você estiver usando o eksctl, use o exemplo a seguir para criar uma função IAM para conta de serviço no EKS:

```
eksctl create iamserviceaccount --name trident-controller --namespace
trident \
  --cluster <my-cluster> --role-name AmazonEKS_FSxN_CSI_DriverRole
--role-only \
  --attach-policy-arn <IAM-Policy ARN> --approve
```

Instale Trident

Trident simplifica o gerenciamento de storage do Amazon FSx for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos.

Você pode instalar Trident usando um dos seguintes métodos:

- Helm
- Complemento do EKS

Se você deseja utilizar a funcionalidade de instantâneo, instale o complemento do controlador de instantâneo CSI. Consulte "Ative a funcionalidade de instantâneo para volumes CSI" para mais informações.

Instale Trident via helm

Identidade do Pod

1. Adicione o repositório Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Instale Trident usando o seguinte exemplo:

```
helm install trident-operator netapp-trident/trident-operator --version 100.2502.1 --namespace trident --create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes da instalação, como nome, namespace, chart, status, versão do aplicativo e número da revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                           APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2502.0    25.02.0
```

Associação de conta de serviço (IRSA)

1. Adicione o repositório Trident:

```
helm repo add netapp-trident https://netapp.github.io/trident-helm-chart
```

2. Defina os valores para **provedor de nuvem** e **identidade de nuvem**:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1  \
--set cloudProvider="AWS" \
--set cloudIdentity="'eks.amazonaws.com/role-arn:
arn:aws:iam::<accountID>:role/<AmazonEKS_FSxN_CSI_DriverRole>'" \
--namespace trident \
--create-namespace
```

Você pode usar o `helm list` comando para revisar detalhes da instalação, como nome, namespace, chart, status, versão do aplicativo e número da revisão.

```
helm list -n trident
```

```
NAME                NAMESPACE   REVISION    UPDATED
STATUS       CHART                          APP VERSION

trident-operator    trident     1           2024-10-14
14:31:22.463122 +0300 IDT    deployed     trident-operator-
100.2510.0    25.10.0
```

Se você pretende usar iSCSI, certifique-se de que o iSCSI esteja habilitado em sua máquina cliente. Se você estiver usando o sistema operacional AL2023 Worker node, é possível automatizar a instalação do cliente iSCSI adicionando o parâmetro node prep na instalação do helm:

```
helm install trident-operator netapp-trident/trident-operator
--version 100.2502.1 --namespace trident --create-namespace --
set nodePrep={iscsi}
```

Instale o Trident via o add-on EKS

O Trident EKS add-on inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O EKS add-on permite garantir de forma consistente que seus clusters Amazon EKS estejam seguros e estáveis e reduz a quantidade de trabalho necessária para instalar, configurar e atualizar add-ons.

Pré-requisitos

Certifique-se de ter o seguinte antes de configurar o add-on Trident para AWS EKS:

- Uma conta de cluster Amazon EKS com assinatura adicional
- Permissões da AWS para o AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos Amazon FSx for NetApp ONTAP existente

Habilite o complemento Trident para AWS

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster para o qual deseja configurar o complemento Trident CSI da NetApp.
4. Selecione **Complementos** e depois selecione **Obter mais complementos**.
5. Siga estes passos para selecionar o software complementar:
   a. Desça a página até a seção **Complementos do AWS Marketplace** e digite **"Trident"** na caixa de pesquisa.
   b. Selecione a caixa de seleção no canto superior direito da caixa Trident by NetApp.
   c. Selecione **Next**.
6. Na página de configurações **Configurar add-ons selecionados**, faça o seguinte:

   Ignore estas etapas se estiver usando a associação de Pod Identity.

   a. Selecione a **Version** que deseja usar.
   b. Se você estiver usando autenticação IRSA, certifique-se de definir os valores de configuração disponíveis nas configurações opcionais:
      - Selecione a **Version** que deseja usar.
      - Siga o **esquema de configuração do complemento** e defina o parâmetro **configurationValues** na seção **Valores de configuração** para o role-arn que você criou na etapa anterior (o valor deve estar no seguinte formato):

```
{

  "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
  "cloudProvider": "AWS"

}
```

   +
   Se você selecionar Override como método de resolução de conflitos, uma ou mais configurações do add-on existente poderão ser substituídas pelas configurações do add-on do Amazon EKS. Se você não habilitar esta opção e houver um conflito com suas configurações existentes, a operação falhará. Você pode usar a mensagem de erro resultante para solucionar o conflito. Antes de selecionar esta opção, certifique-se de que o add-on do Amazon EKS não gerencie configurações que você precise gerenciar manualmente.

7. Escolha **Próximo**.
8. Na página **Revisar e adicionar**, escolha **Criar**.

   Após a conclusão da instalação do complemento, você verá o complemento instalado.

AWS CLI

1. Crie o `add-on.json` arquivo:

Para Pod Identity, utilize o seguinte formato:

Use o

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
}
```

Para autenticação IRSA, utilize o seguinte formato:

```
{
  "clusterName": "<eks-cluster>",
  "addonName": "netapp_trident-operator",
  "addonVersion": "v25.6.0-eksbuild.1",
  "serviceAccountRoleArn": "<role ARN>",
  "configurationValues": {
    "cloudIdentity": "'eks.amazonaws.com/role-arn: <role ARN>'",
    "cloudProvider": "AWS"
  }
}
```

Substitua `<role ARN>` pelo ARN da função que foi criada na etapa anterior.

2. Instale o complemento Trident EKS.

```
aws eks create-addon --cli-input-json file://add-on.json
```

eksctl

O seguinte comando de exemplo instala o Trident EKS add-on:

```
eksctl create addon --name netapp_trident-operator --cluster
<cluster_name> --force
```

Atualize o complemento Trident EKS

Console de gerenciamento

1. Abra o console do Amazon EKS https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster para o qual deseja atualizar o software complementar NetApp Trident CSI.
4. Selecione a guia **Add-ons**.
5. Selecione **Trident por NetApp** e depois selecione **Editar**.
6. Na página **Configurar Trident por NetApp**, faça o seguinte:
   a. Selecione a **Version** que deseja usar.
   b. Expanda as **Configurações opcionais de configuração** e modifique conforme necessário.
   c. Selecione **Save changes**.

AWS CLI

O exemplo a seguir atualiza o add-on EKS:

```
aws eks update-addon --cluster-name <eks_cluster_name> --addon-name
netapp_trident-operator --addon-version v25.6.0-eksbuild.1 \
  --service-account-role-arn <role-ARN> --resolve-conflict preserve \
  --configuration-values "{\"cloudIdentity\":
\"'eks.amazonaws.com/role-arn: <role ARN>'\"}"
```

eksctl

- Verifique a versão atual do seu FSxN Trident CSI add-on. Substitua `my-cluster` pelo nome do seu cluster.

```
eksctl get addon --name netapp_trident-operator --cluster my-cluster
```

Exemplo de saída:

```
NAME                          VERSION              STATUS    ISSUES
IAMROLE    UPDATE AVAILABLE    CONFIGURATION VALUES
netapp_trident-operator    v25.6.0-eksbuild.1    ACTIVE    0
{"cloudIdentity":"'eks.amazonaws.com/role-arn:
arn:aws:iam::139763910815:role/AmazonEKS_FSXN_CSI_DriverRole'"}
```

- Atualize o software complementar para a versão retornada em UPDATE AVAILABLE na saída da etapa anterior.

```
eksctl update addon --name netapp_trident-operator --version
v25.6.0-eksbuild.1 --cluster my-cluster --force
```

Se você remover a `--force` opção e alguma das configurações do add-on do Amazon EKS entrar em conflito com suas configurações existentes, a atualização do add-on do Amazon EKS falhará; você receberá uma mensagem de erro para ajudá-lo a resolver o conflito. Antes de especificar esta opção, certifique-se de que o add-on do Amazon EKS não gerencie configurações que você precisa gerenciar, pois essas configurações serão sobrescritas com esta opção. Para mais informações sobre outras opções para esta configuração, consulte "Complementos". Para mais informações sobre o gerenciamento de campos do Kubernetes no Amazon EKS, consulte "Gerenciamento de campos do Kubernetes".

Desinstalar/remover o Trident EKS add-on

Você tem duas opções para remover um add-on do Amazon EKS:

- **Preservar software complementar no seu cluster** – Esta opção remove o gerenciamento de quaisquer configurações pelo Amazon EKS. Ela também remove a capacidade do Amazon EKS de notificá-lo sobre atualizações e atualizar automaticamente o add-on do Amazon EKS após você iniciar uma atualização. No entanto, ela preserva o software complementar no seu cluster. Esta opção transforma o add-on em uma instalação autogerenciada, em vez de um add-on do Amazon EKS. Com esta opção, não há tempo de inatividade para o add-on. Mantenha a `--preserve` opção no comando para preservar o add-on.
- **Remova o software complementar completamente do seu cluster** – NetApp recomenda que você remova o complemento do Amazon EKS do seu cluster somente se não houver recursos no seu cluster que dependam dele. Remova a `--preserve` opção do comando `delete` para remover o software complementar.

Se o software complementar tiver uma conta IAM associada a ele, a conta IAM não será removida.

Console de gerenciamento

1. Abra o console do Amazon EKS em https://console.aws.amazon.com/eks/home#/clusters.
2. No painel de navegação à esquerda, selecione **Clusters**.
3. Selecione o nome do cluster do qual você deseja remover o complemento NetApp Trident CSI.
4. Selecione a aba **Complementos** e depois selecione **Trident by NetApp**.*
5. Selecione **Remove**.
6. Na caixa de diálogo **Remover netapp_trident-operator confirmation**, faça o seguinte:
   a. Se você deseja que o Amazon EKS pare de gerenciar as configurações do software complementar, selecione **Preservar no cluster**. Faça isso se quiser manter o software complementar no seu cluster para que você possa gerenciar todas as configurações do software complementar por conta própria.
   b. Digite **netapp_trident-operator**.
   c. Selecione **Remove**.

AWS CLI

Substitua `my-cluster` pelo nome do seu cluster e, em seguida, execute o seguinte comando.

```
aws eks delete-addon --cluster-name my-cluster --addon-name
netapp_trident-operator --preserve
```

eksctl

O comando a seguir desinstala o Trident EKS add-on:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Configurar o backend de armazenamento

Integração do driver ONTAP SAN e NAS

Para criar um backend de armazenamento, você precisa criar um arquivo de configuração em formato JSON ou YAML. O arquivo deve especificar o tipo de armazenamento que você deseja (NAS ou SAN), o sistema de arquivos, a SVM de onde obtê-lo e como autenticar com ela. O exemplo a seguir mostra como definir um armazenamento baseado em NAS e usar um segredo da AWS para armazenar as credenciais da SVM que você deseja usar:

YAML

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-nas
  namespace: trident
spec:
  version: 1
  storageDriverName: ontap-nas
  backendName: tbc-ontap-nas
  svm: svm-name
  aws:
    fsxFilesystemID: fs-xxxxxxxxxx
  credentials:
    name: "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name"
    type: awsarn
```

JSON

```
{
  "apiVersion": "trident.netapp.io/v1",
  "kind": "TridentBackendConfig",
  "metadata": {
    "name": "backend-tbc-ontap-nas"
    "namespace": "trident"
  },
  "spec": {
    "version": 1,
    "storageDriverName": "ontap-nas",
    "backendName": "tbc-ontap-nas",
    "svm": "svm-name",
    "aws": {
      "fsxFilesystemID": "fs-xxxxxxxxxx"
    },
    "managementLIF": null,
    "credentials": {
      "name": "arn:aws:secretsmanager:us-west-2:xxxxxxxx:secret:secret-
name",
      "type": "awsarn"
    }
  }
}
```

Execute os seguintes comandos para criar e validar a Trident Backend Configuration (TBC):

- Crie a configuração de backend do Trident (TBC) a partir do arquivo yaml e execute o seguinte comando:

```
kubectl create -f backendconfig.yaml -n trident
```

```
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-nas created
```

- Valide se a configuração do backend Trident (TBC) foi criada com sucesso:

```
Kubectl get tbc -n trident
```

```
NAME                           BACKEND NAME          BACKEND UUID
PHASE   STATUS

backend-tbc-ontap-nas          tbc-ontap-nas         933e0071-66ce-4324-
b9ff-f96d916ac5e9   Bound   Success
```

Detalhes do driver FSx for ONTAP

Você pode integrar Trident com Amazon FSx for NetApp ONTAP usando os seguintes drivers:

- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio volume Amazon FSx for NetApp ONTAP. Recomendado para armazenamento em bloco.
- `ontap-nas`: Cada PV provisionado é um volume completo do Amazon FSx for NetApp ONTAP. Recomendado para NFS e SMB.
- `ontap-san-economy`: Cada PV provisionado é um LUN com um número configurável de LUNs por Amazon FSx for NetApp ONTAP volume.
- `ontap-nas-economy`: Cada PV provisionado é um qtree, com um número configurável de qtrees por Amazon FSx for NetApp ONTAP volume.
- `ontap-nas-flexgroup`: Cada PV provisionado é um volume completo do Amazon FSx for NetApp ONTAP FlexGroup.

Para obter detalhes sobre o driver, consulte "Drivers NAS" e "Drivers SAN".

Depois que o arquivo de configuração for criado, execute este comando para criá-lo no seu EKS:

```
kubectl create -f configuration_file
```

Para verificar o status, execute este comando:

```
kubectl get tbc -n trident
```

```
NAME                    BACKEND NAME            BACKEND UUID
PHASE   STATUS
backend-fsx-ontap-nas   backend-fsx-ontap-nas   7a551921-997c-4c37-a1d1-
f2f4c87fa629   Bound   Success
```

Configuração avançada do backend e exemplos

Consulte a tabela a seguir para as opções de configuração do backend:

Parâmetro	Descrição	Exemplo
`version`		Sempre 1
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`
`backendName`	Nome personalizado ou o storage backend	Nome do driver + "_" + dataLIF
`managementLIF`	Endereço IP de um cluster ou LIF de gerenciamento de SVM. Um nome de domínio totalmente qualificado (FQDN) pode ser especificado. Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o parâmetro IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555]. Se você fornecer o `fsxFilesystemID` sob o campo `aws`, não é necessário fornecer o `managementLIF`, pois Trident recupera as informações da SVM `managementLIF` da AWS. Portanto, você deve fornecer credenciais para um usuário na SVM (por exemplo: vsadmin) e o usuário deve ter a função `vsadmin`.	"10.0.0.1", "[2001:1234:abcd::fefe]"

Parâmetro	Descrição	Exemplo
`dataLIF`	Endereço IP do protocolo LIF. **ONTAP NAS drivers**: NetApp recomenda especificar dataLIF. Caso não seja fornecido, Trident busca os dataLIFs no SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Pode ser alterado após a configuração inicial. Consulte . **ONTAP SAN drivers**: Não especifique para iSCSI. Trident usa ONTAP Selective LUN Map para descobrir os LIFs iSCSI necessários para estabelecer uma sessão multipath. Um aviso é gerado se dataLIF for definido explicitamente. Pode ser configurado para usar endereços IPv6 se Trident foi instalado usando o sinalizador IPv6. Os endereços IPv6 devem ser definidos entre colchetes, como [28e8:d9fb:a825:b7bf:69a8:d02f:9e7b:3555].	
`autoExportPolicy`	Ativar a criação e atualização automática de políticas de exportação [Booleano]. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	`false`
`autoExportCIDRs`	Lista de CIDRs para filtrar os IPs dos nós do Kubernetes quando `autoExportPolicy` estiver habilitado. Usando as opções `autoExportPolicy` e `autoExportCIDRs`, Trident pode gerenciar políticas de exportação automaticamente.	"["0.0.0.0/0", "::/0"]"
`labels`	Conjunto de rótulos arbitrários formatados em JSON para aplicar aos volumes	""
`clientCertificate`	Valor codificado em Base64 do certificado do cliente. Usado para autenticação baseada em certificado	""

Parâmetro	Descrição	Exemplo
`clientPrivateKey`	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado	""
`trustedCACertificate`	Valor codificado em Base64 do certificado da CA confiável. Opcional. Usado para autenticação baseada em certificado.	""
`username`	Nome de usuário para conectar-se ao cluster ou SVM. Usado para autenticação baseada em credenciais. Por exemplo, vsadmin.	
`password`	Senha para conectar-se ao cluster ou SVM. Usada para autenticação baseada em credenciais.	
`svm`	Máquina virtual de storage para usar	Derivado se um SVM managementLIF for especificado.
`storagePrefix`	Prefixo usado ao provisionar novos volumes no SVM. Não pode ser modificado após a criação. Para atualizar este parâmetro, você precisará criar um novo backend.	`trident`
`limitAggregateUsage`	**Não especifique para Amazon FSx para NetApp ONTAP.** As configurações fornecidas `fsxadmin` e `vsadmin` não contêm as permissões necessárias para recuperar o uso agregado e limitá-lo usando Trident.	Não use.
`limitVolumeSize`	O provisionamento falha se o tamanho do volume solicitado for superior a este valor. Também restringe o tamanho máximo dos volumes que gerencia para qtrees e LUNs, e a `qtreesPerFlexvol` opção permite personalizar o número máximo de qtrees por FlexVol volume	"" (não aplicado por padrão)
`lunsPerFlexvol`	LUNs máximas por FlexVol volume, deve estar no intervalo [50, 200]. Somente SAN.	“100”

Parâmetro	Descrição	Exemplo
`debugTraceFlags`	Sinalizadores de depuração para usar na resolução de problemas. Exemplo, {"api":false, "method":true} não use `debugTraceFlags` a menos que esteja solucionando problemas e precise de um despejo de log detalhado.	null
`nfsMountOptions`	Lista separada por vírgulas de opções de montagem NFS. As opções de montagem para volumes persistentes do Kubernetes são normalmente especificadas nas classes de armazenamento, mas se nenhuma opção de montagem for especificada em uma classe de armazenamento, Trident usará as opções de montagem especificadas no arquivo de configuração do backend de storage. Se nenhuma opção de montagem for especificada na classe de armazenamento ou no arquivo de configuração, Trident não definirá nenhuma opção de montagem em um volume persistente associado.	""
`nasType`	Configurar a criação de volumes NFS ou SMB. As opções são `nfs`, `smb` ou null. **Deve ser definido como `smb` para volumes SMB.** Definir como null define volumes NFS por padrão.	`nfs`
`qtreesPerFlexvol`	Número máximo de qtrees por FlexVol volume, deve estar no intervalo [50, 300]	`"200"`
`smbShare`	Você pode especificar uma das seguintes opções: o nome de um compartilhamento SMB criado usando o Microsoft Management Console ou ONTAP CLI, ou um nome para permitir que Trident crie o compartilhamento SMB. Este parâmetro é obrigatório para Amazon FSx for ONTAP backends.	`smb-share`

Parâmetro	Descrição	Exemplo
`useREST`	Parâmetro booleano para usar as ONTAP REST APIs. Quando definido como `true`, Trident usará as ONTAP REST APIs para se comunicar com o backend. Este recurso requer ONTAP 9.11.1 e versões posteriores. Além disso, a função de login do ONTAP utilizada deve ter acesso ao aplicativo `ontap`. Isso é atendido pelas funções predefinidas `vsadmin` e `cluster-admin`.	`false`
`aws`	Você pode especificar o seguinte no arquivo de configuração do AWS FSx para ONTAP: - `fsxFilesystemID`: especificar o ID do sistema de arquivos AWS FSx. - `apiRegion`: nome da região da API da AWS. - `apikey`: chave da API da AWS. - `secretKey`: chave secreta da AWS.	`""` `""` `""`
`credentials`	Especifique as credenciais do FSx SVM a serem armazenadas no AWS Secrets Manager. - `name`: Amazon Resource Name (ARN) do segredo, que contém as credenciais do SVM. - `type`: Defina como `awsarn`. Consulte "Crie um segredo do AWS Secrets Manager" para mais informações.	

Opções de configuração de backend para provisionamento de volumes

Você pode controlar o provisionamento padrão usando essas opções na seção `defaults` do arquivo de configuração. Para um exemplo, veja os exemplos de configuração abaixo.

Parâmetro	Descrição	Padrão
`spaceAllocation`	Alocação de espaço para LUNs	`true`
`spaceReserve`	Modo de reserva de espaço; "none" (fino) ou "volume" (grosso)	`none`
`snapshotPolicy`	Política do Snapshot a ser usada	`none`

Parâmetro	Descrição	Padrão
`qosPolicy`	Grupo de políticas de QoS a ser atribuído aos volumes criados. Escolha um dos qosPolicy ou adaptiveQosPolicy por pool de storage ou backend. O uso de grupos de políticas de QoS com Trident requer ONTAP 9.8 ou posterior. Você deve usar um grupo de políticas de QoS não compartilhado e garantir que o grupo de políticas seja aplicado a cada componente individualmente. Um grupo de políticas de QoS compartilhado impõe o limite máximo para a taxa de transferência total de todas as cargas de trabalho.	""
`adaptiveQosPolicy`	Grupo de políticas de QoS adaptável para atribuir aos volumes criados. Escolha uma das opções qosPolicy ou adaptiveQosPolicy por pool de storage ou backend. Não compatível com ontap-nas-economy.	""
`snapshotReserve`	Porcentagem do volume reservada para snapshots "0"	Se `snapshotPolicy` for `none`, `else` ""
`splitOnClone`	Separar um clone de seu progenitor no momento da criação	`false`
`encryption`	Habilite NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se a NAE estiver habilitada no backend, qualquer volume provisionado no Trident terá a NAE habilitada. Para mais informações, consulte: "Como Trident funciona com NVE e NAE".	`false`
`luksEncryption`	Ative LUKS criptografia. Consulte "Use Linux Unified Key Setup (LUKS)". Apenas SAN.	""
`tieringPolicy`	Política de tiering a ser usada `none`	
`unixPermissions`	Modo para novos volumes. **Deixe em branco para volumes SMB.**	""

Parâmetro	Descrição	Padrão
`securityStyle`	Estilo de segurança para novos volumes. NFS suporta `mixed` e `unix` estilos de segurança. SMB suporta `mixed` e `ntfs` estilos de segurança.	NFS default é `unix`. SMB default é `ntfs`.

Provisionar volumes SMB

Você pode provisionar volumes SMB usando o `ontap-nas` driver. Antes de concluir Integração do driver ONTAP SAN e NAS, execute estas etapas: "Prepare-se para provisionar volumes SMB".

Configure uma classe de armazenamento e um PVC

Configure um objeto StorageClass do Kubernetes e crie a classe de armazenamento para instruir Trident sobre como provisionar volumes. Crie um PersistentVolumeClaim (PVC) que utilize o StorageClass do Kubernetes configurado para solicitar acesso ao PV. Você pode então montar o PV em um pod.

Crie uma storage class

Configure um objeto StorageClass do Kubernetes

O "Objeto Kubernetes StorageClass" objeto identifica Trident como o provisionador usado para essa classe e instrui Trident sobre como provisionar um volume. Use este exemplo para configurar Storageclass para volumes usando NFS (consulte a seção de atributos do Trident abaixo para obter a lista completa de atributos):

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  provisioningType: "thin"
  snapshots: "true"
```

Use este exemplo para configurar Storageclass para volumes usando iSCSI:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
  provisioningType: "thin"
  snapshots: "true"
```

Para provisionar volumes NFSv3 no AWS Bottlerocket, adicione o `mountOptions`necessário à classe de armazenamento:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-nas"
  media: "ssd"
  provisioningType: "thin"
  snapshots: "true"
mountOptions:
  - nfsvers=3
  - nolock
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Crie uma storage class

Passos

1. Este é um objeto do Kubernetes, portanto, use `kubectl` para criá-lo no Kubernetes.

```
kubectl create -f storage-class-ontapnas.yaml
```

2. Agora você deverá ver uma classe de storage **basic-csi** tanto no Kubernetes quanto no Trident, e o Trident deverá ter descoberto os pools no backend.

```
kubectl get sc basic-csi
```

```
NAME          PROVISIONER             AGE
basic-csi     csi.trident.netapp.io   15h
```

Crie o PVC

Um "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster.

O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o PVC, você pode montar o volume em um pod.

Exemplos de manifestos

Manifestos de exemplo de PersistentVolumeClaim

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWX

Este exemplo mostra um PVC básico com acesso RWX que está associado a um StorageClass chamado `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteMany
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-gold
```

Exemplo de PVC usando iSCSI

Este exemplo mostra um PVC básico para iSCSI com acesso RWO que está associado a um StorageClass chamado `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 1Gi
storageClassName: protection-gold
```

Criar PVC

Passos

1. Crie o PVC.

```
kubectl create -f pvc.yaml
```

2. Verifique o status do PVC.

```
kubectl get pvc
```

```
NAME          STATUS VOLUME     CAPACITY ACCESS MODES STORAGECLASS AGE
pvc-storage Bound  pv-name 2Gi      RWO                           5m
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Atributos do Trident

Esses parâmetros determinam quais pools de storage gerenciados pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
mídia[1]	string	hdd, híbrido, ssd	O pool contém mídias deste tipo; híbrido significa ambos	Tipo de mídia especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	string	fino, grosso	Pool suporta este método de provisionamento	Método de provisionamento especificado	espesso: all ontap; fino: all ontap & solidfire-san
backendType	string	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertence a este tipo de backend	Backend especificado	Todos os drivers
instantâneos	bool	true, false	O pool suporta volumes com snapshots	Volume com snapshots ativados	ontap-nas, ontap-san, solidfire-san
clones	bool	true, false	Pool suporta clonagem de volumes	Volume com clonagem ativada	ontap-nas, ontap-san, solidfire-san

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
criptografia	bool	true, false	Pool suporta volumes criptografados	Volume com criptografia ativada	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san
IOPS	inteiro	inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantiu esses IOPS	solidfire-san

[1]: não suportado pelos sistemas ONTAP Select

Implantar aplicação de exemplo

Após a criação da classe de armazenamento e do PVC, você pode montar o PV em um pod. Esta seção lista o comando de exemplo e a configuração para anexar o PV a um pod.

Passos

1. Monte o volume em um pod.

```
kubectl create -f pv-pod.yaml
```

Estes exemplos mostram configurações básicas para conectar o PVC a um pod: **configuração básica**:

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: pv-storage
      persistentVolumeClaim:
       claimName: basic
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: pv-storage
```

Você pode monitorar o progresso usando `kubectl get pod --watch`.

2. Verifique se o volume está montado em `/my/mount/path`.

```
kubectl exec -it pv-pod -- df -h /my/mount/path
```

```
Filesystem                                                      Size
Used  Avail Use%  Mounted on
192.168.188.78:/trident_pvc_ae45ed05_3ace_4e7c_9080_d2a83ae03d06  1.1G
320K  1.0G   1%   /my/mount/path
```

Agora você pode excluir o Pod. O aplicativo Pod deixará de existir, mas o volume permanecerá.

```
kubectl delete pod pv-pod
```

Configurar o complemento Trident EKS em um cluster EKS

NetApp Trident simplifica o gerenciamento de storage do Amazon FSx for NetApp ONTAP no Kubernetes para permitir que seus desenvolvedores e administradores se concentrem na implantação de aplicativos. O NetApp Trident EKS add-on inclui os patches de segurança mais recentes, correções de bugs e é validado pela AWS para funcionar com o Amazon EKS. O EKS add-on permite garantir de forma consistente que seus clusters Amazon EKS estejam seguros e estáveis e reduz a quantidade de trabalho necessária para instalar, configurar e atualizar add-ons.

Pré-requisitos

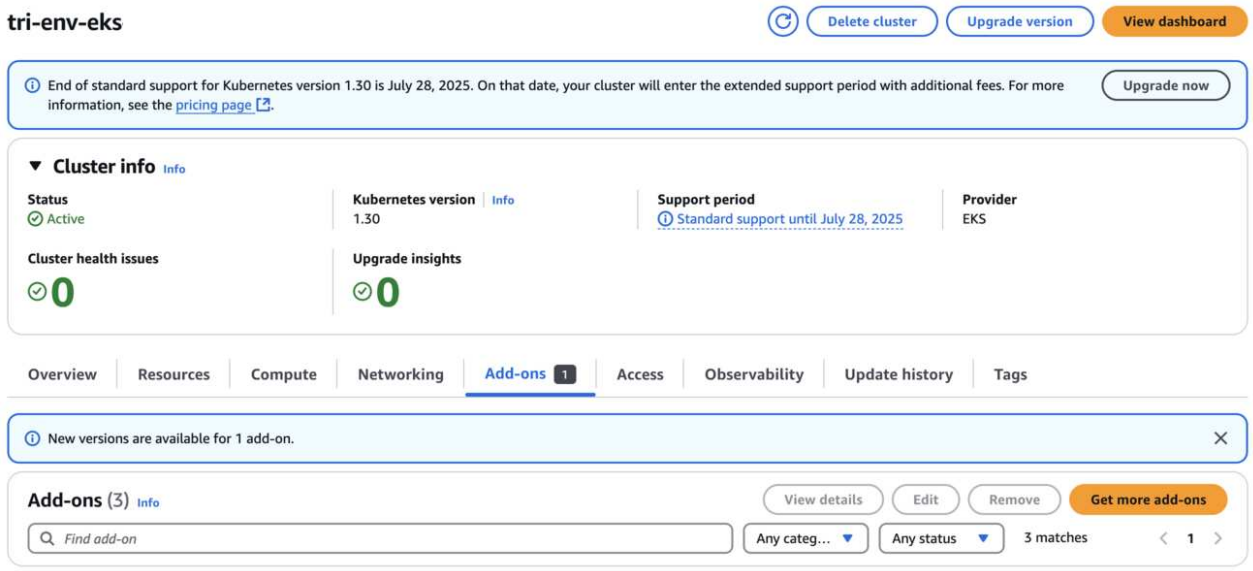
Certifique-se de ter o seguinte antes de configurar o add-on Trident para AWS EKS:

- Uma conta de cluster Amazon EKS com permissões para trabalhar com complementos. Consulte "Complementos do Amazon EKS".
- Permissões da AWS para o AWS marketplace:
  `"aws-marketplace:ViewSubscriptions",`
  `"aws-marketplace:Subscribe",`
  `"aws-marketplace:Unsubscribe`
- Tipo de AMI: Amazon Linux 2 (AL2_x86_64) ou Amazon Linux 2 Arm(AL2_ARM_64)
- Tipo de nó: AMD ou ARM
- Um sistema de arquivos Amazon FSx for NetApp ONTAP existente

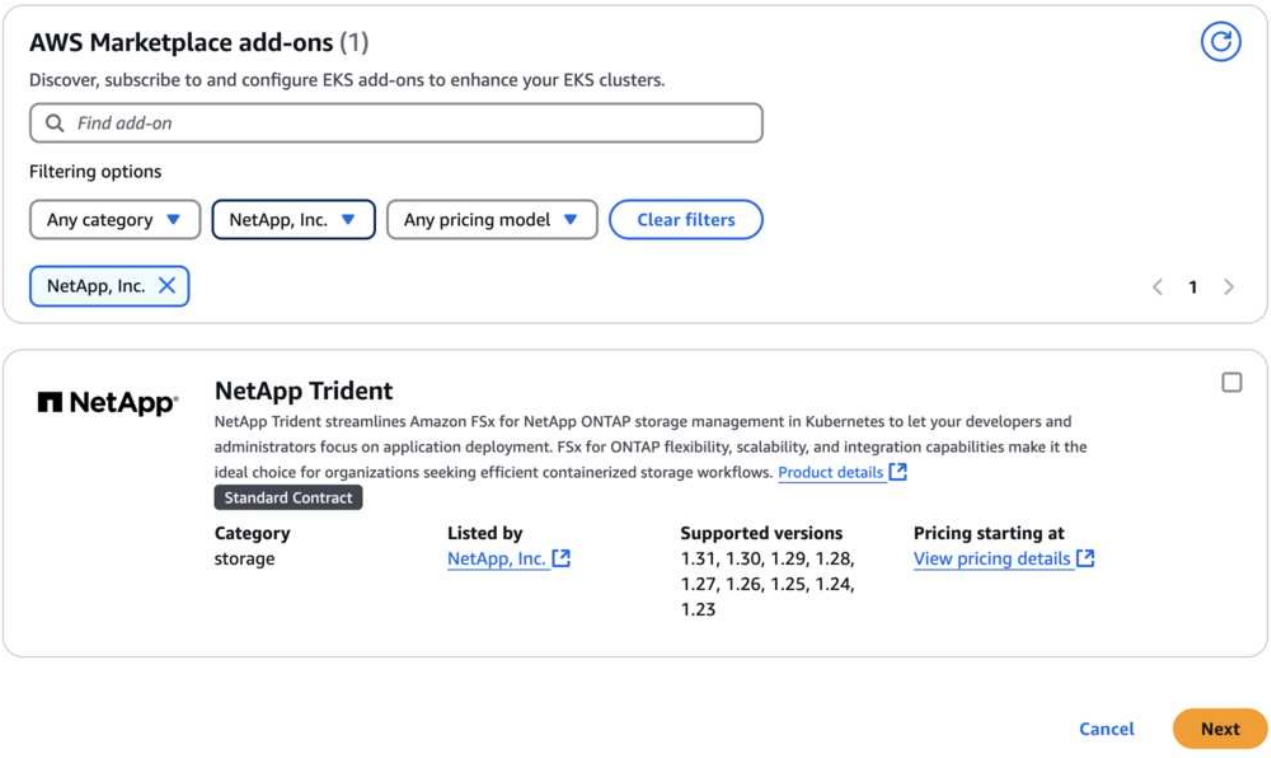
Passos

1. Certifique-se de criar uma função do IAM e um segredo da AWS para permitir que os pods do EKS acessem os recursos da AWS. Para obter instruções, consulte "Crie uma função do IAM e um segredo da AWS".

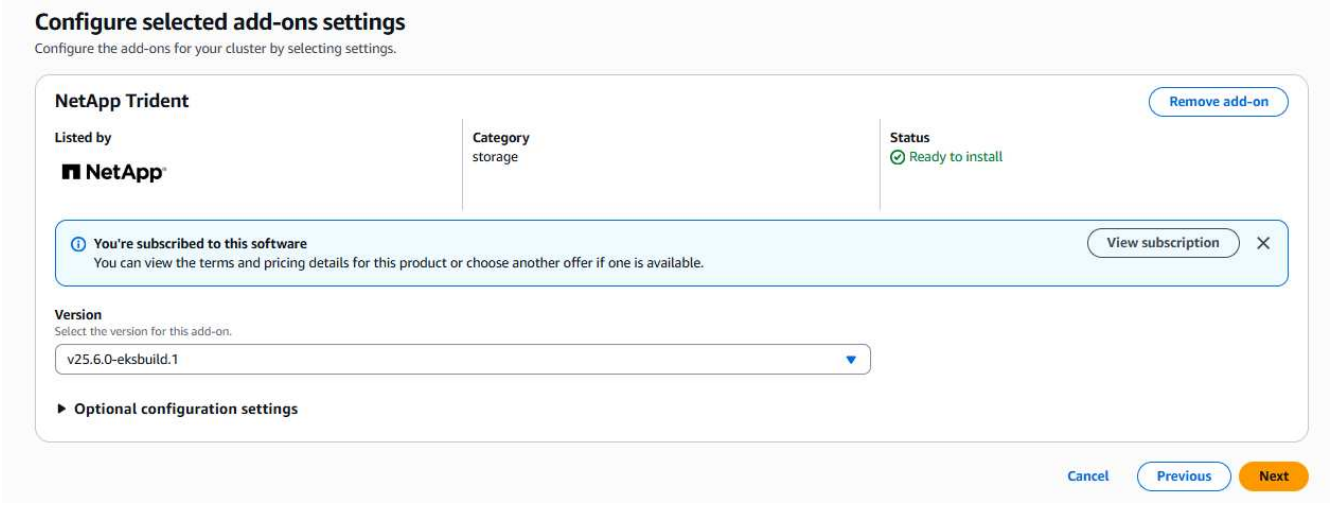
2. No seu cluster Kubernetes EKS, navegue até a guia **Add-ons**.



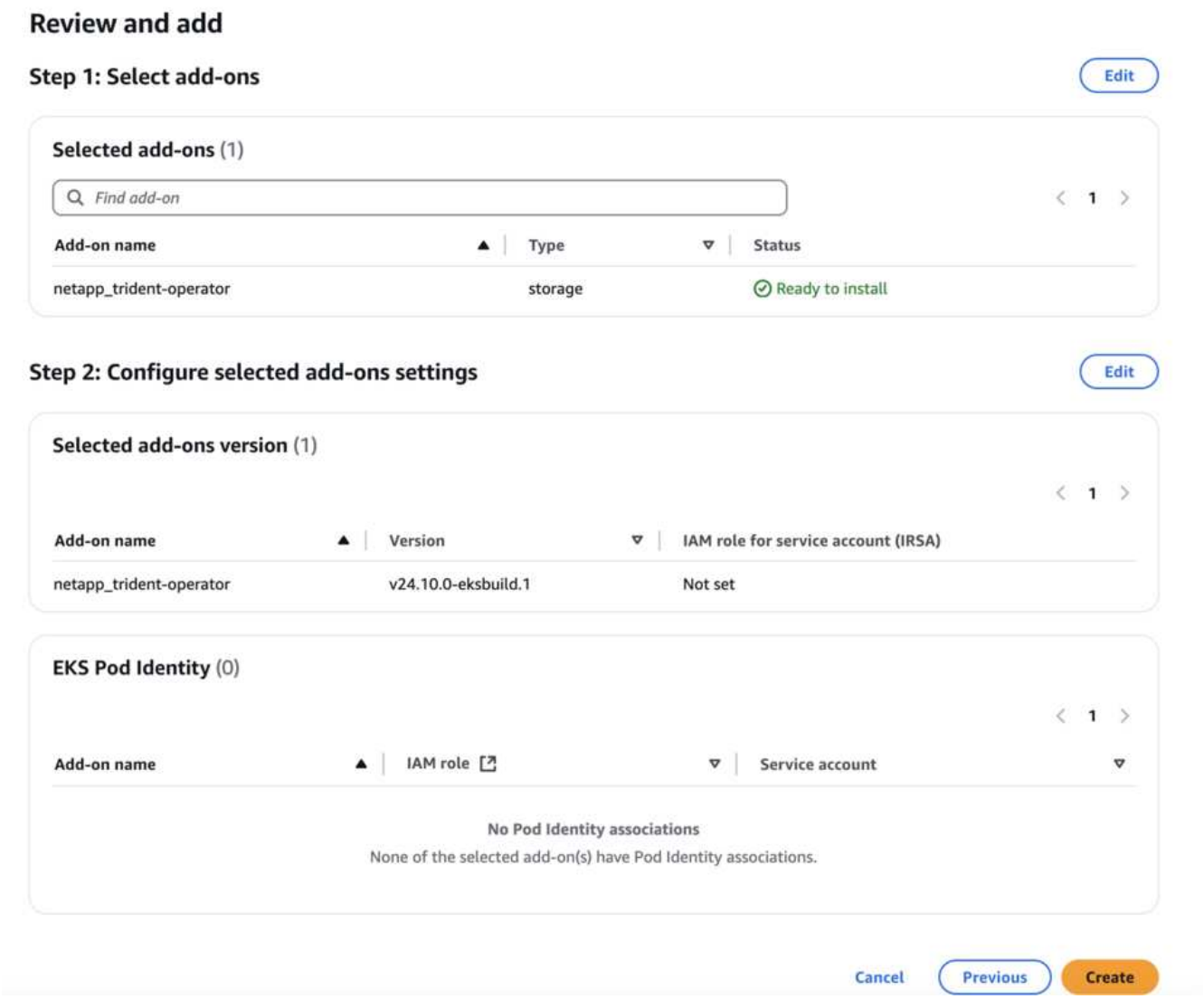
3. Acesse **AWS Marketplace add-ons** e escolha a categoria *storage*.



4. Localize **NetApp Trident** e selecione a caixa de seleção para o add-on Trident e clique em **Próximo**.
5. Escolha a versão desejada do add-on.

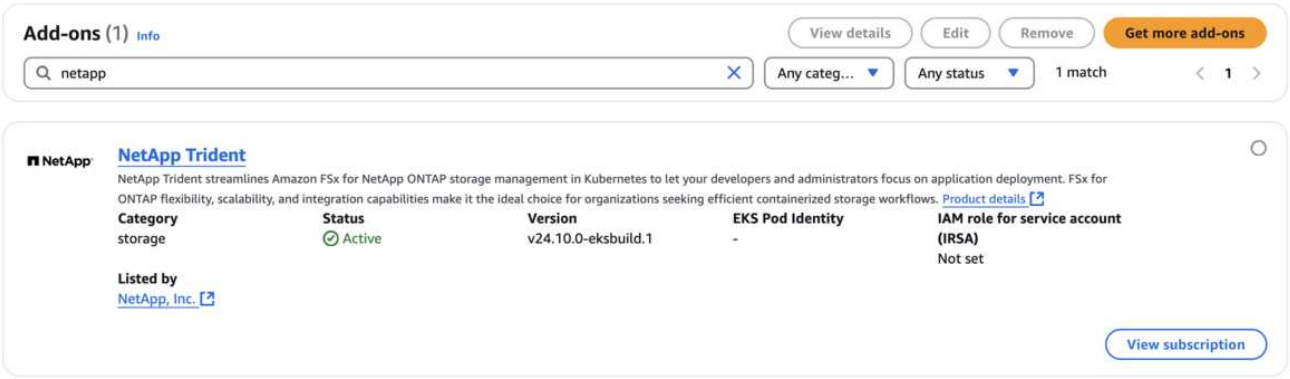


6. Configurar as definições adicionais necessárias.



7. Se você estiver usando IRSA (IAM roles para conta de serviço), consulte as etapas de configuração adicionais "aqui".
8. Selecione **Create**.

9. Verifique se o status do add-on é *Ativo*.



10. Execute o seguinte comando para verificar se Trident está instalado corretamente no cluster:

```
kubectl get pods -n trident
```

11. Continue a configuração e configure o backend de storage. Para obter informações, consulte "Configurar o backend de armazenamento".

Instalar/desinstalar o Trident EKS add-on usando CLI

Instale o NetApp Trident EKS add-on usando CLI:

O seguinte comando de exemplo instala o Trident EKS add-on:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.0-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.1:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.1-eksbuild.1` (com uma versão dedicada)

O seguinte comando de exemplo instala o Trident EKS add-on versão 25.6.2:
`eksctl create addon --cluster clusterName --name netapp_trident-operator --version v25.6.2-eksbuild.1` (com uma versão dedicada)

Desinstale o complemento NetApp Trident EKS usando a CLI:

O comando a seguir desinstala o Trident EKS add-on:

```
eksctl delete addon --cluster K8s-arm --name netapp_trident-operator
```

Criar backends com kubectl

Um backend define a relação entre Trident e um sistema de storage. Ele informa ao Trident como se comunicar com esse sistema de storage e como o Trident deve provisionar volumes a partir dele. Após a instalação do Trident, o próximo passo é criar um backend. A `TridentBackendConfig` Custom Resource Definition (CRD) permite que você crie e gerencie backends do Trident diretamente pela interface do Kubernetes.

Você pode fazer isso usando `kubectl` ou a ferramenta de linha de comando equivalente para sua distribuição do Kubernetes.

`TridentBackendConfig`

`TridentBackendConfig` (`tbc`, `tbconfig`, `tbackendconfig`) é um CRD de frontend com namespace que permite gerenciar backends do Trident usando `kubectl`. Administradores de Kubernetes e de storage agora podem criar e gerenciar backends diretamente por meio da CLI do Kubernetes sem a necessidade de um utilitário de linha de comando dedicado (`tridentctl`).

Ao criar um `TridentBackendConfig` objeto, ocorre o seguinte:

- Um backend é criado automaticamente pelo Trident com base na configuração que você fornece. Isso é representado internamente como um `TridentBackend` (`tbe`, `tridentbackend`) CR.
- O `TridentBackendConfig` está exclusivamente ligado a um `TridentBackend` que foi criado pela Trident.

Cada `TridentBackendConfig` mantém um mapeamento um-para-um com um `TridentBackend`. O primeiro é a interface fornecida ao usuário para projetar e configurar backends; o segundo é como Trident representa o objeto backend real.

> `TridentBackend` Os CRs são criados automaticamente pelo Trident. Você **não deve** modificá-los. Se você quiser fazer atualizações nos backends, faça isso modificando o `TridentBackendConfig` objeto.

Veja o exemplo a seguir para o formato do `TridentBackendConfig` CR:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Você também pode consultar os exemplos no "trident-installer" diretório para obter configurações de amostra para a plataforma de storage/serviço desejado.

O `spec` aceita parâmetros de configuração específicos do backend. Neste exemplo, o backend usa o `ontap-san` driver de storage e utiliza os parâmetros de configuração que estão tabulados aqui. Para a lista de opções de configuração para o driver de storage desejado, consulte o "Informações de configuração do backend para o seu driver de storage".

A `spec` seção também inclui `credentials` e `deletionPolicy` campos, que são recém-introduzidos no `TridentBackendConfig` CR:

- `credentials`: Este parâmetro é obrigatório e contém as credenciais usadas para autenticação com o sistema de storage/serviço. Isso é definido como um segredo do Kubernetes criado pelo usuário. As credenciais não podem ser passadas em texto simples e resultarão em um erro.
- `deletionPolicy`: Este campo define o que deve acontecer quando o `TridentBackendConfig` for excluído. Ele pode assumir um dos dois valores possíveis:
  - `delete`: Isso resulta na exclusão tanto do `TridentBackendConfig`CR quanto do backend associado. Este é o valor padrão.
  - `retain`: Quando uma `TridentBackendConfig` CR é excluída, a definição do backend ainda estará presente e poderá ser gerenciada com `tridentctl`. Definir a política de exclusão para `retain` permite que os usuários façam downgrade para uma versão anterior (pré-21.04) e mantenham os backends criados. O valor deste campo pode ser atualizado após uma `TridentBackendConfig` CR ser criada.

> O nome de um backend é definido usando `spec.backendName`. Se não for especificado, o nome do backend é definido como o nome do objeto `TridentBackendConfig` (metadata.name). Recomenda-se definir explicitamente os nomes dos backends usando `spec.backendName`.

> Os backends que foram criados com `tridentctl` não possuem um objeto `TridentBackendConfig` associado. Você pode optar por gerenciar esses backends com `kubectl` criando um `TridentBackendConfig` CR. É preciso ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName` e assim por diante). O Trident vinculará automaticamente o novo `TridentBackendConfig` ao backend preexistente.

Visão geral das etapas

Para criar um novo backend usando `kubectl`, você deve fazer o seguinte:

1. Crie um "Kubernetes Secret" segredo. O segredo contém as credenciais que Trident precisa para se comunicar com o cluster/serviço de storage.
2. Crie um `TridentBackendConfig`objeto. Este objeto contém detalhes sobre o cluster/serviço de storage e faz referência ao segredo criado na etapa anterior.

Após criar um backend, você pode observar seu status usando `kubectl get tbc <tbc-name> -n <trident-namespace>` e coletar detalhes adicionais.

Passo 1: criar um segredo do Kubernetes

Crie um segredo que contenha as credenciais de acesso ao backend. Isso é exclusivo para cada serviço/plataforma de storage. Veja um exemplo:

```
kubectl -n trident create -f backend-tbc-ontap-san-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-ontap-san-secret
type: Opaque
stringData:
  username: cluster-admin
  password: password
```

Esta tabela resume os campos que devem ser incluídos no Secret para cada plataforma de storage:

Descrição dos campos secretos da plataforma de storage	Segredo	Descrição dos campos
Azure NetApp Files	clientID	O client ID de um registro de aplicativo
Element (NetApp HCI/SolidFire)	Endpoint	MVIP para o SolidFire cluster com credenciais de locatário
ONTAP	nome de usuário	Nome de usuário para conectar-se ao cluster/SVM. Usado para autenticação baseada em credencial
ONTAP	senha	Senha para conectar-se ao cluster/SVM. Usada para autenticação baseada em credencial
ONTAP	clientPrivateKey	Valor codificado em Base64 da chave privada do cliente. Usado para autenticação baseada em certificado
ONTAP	chapUsername	Nome de usuário de entrada. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`
ONTAP	chapInitiatorSecret	Segredo do iniciador CHAP. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`

Descrição dos campos secretos da plataforma de storage	Segredo	Descrição dos campos
ONTAP	chapTargetUsername	Nome de usuário de destino. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`
ONTAP	chapTargetInitiatorSecret	Segredo do iniciador do alvo CHAP. Obrigatório se useCHAP=true. Para `ontap-san` e `ontap-san-economy`

O segredo criado nesta etapa será referenciado no campo `spec.credentials` do objeto `TridentBackendConfig` que é criado na próxima etapa.

Etapa 2: Criar o `TridentBackendConfig` CR

Agora você está pronto para criar seu `TridentBackendConfig` CR. Neste exemplo, um backend que utiliza o `ontap-san` driver é criado usando o `TridentBackendConfig` objeto mostrado abaixo:

```
kubectl -n trident create -f backend-tbc-ontap-san.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc-ontap-san
spec:
  version: 1
  backendName: ontap-san-backend
  storageDriverName: ontap-san
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  svm: trident_svm
  credentials:
    name: backend-tbc-ontap-san-secret
```

Etapa 3: Verificar o status do `TridentBackendConfig`CR

Agora que você criou o `TridentBackendConfig` CR, pode verificar o status. Veja o exemplo a seguir:

```
kubectl -n trident get tbc backend-tbc-ontap-san
NAME                     BACKEND NAME          BACKEND UUID
PHASE   STATUS
backend-tbc-ontap-san   ontap-san-backend     8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success
```

Um backend foi criado e vinculado com sucesso ao `TridentBackendConfig` CR.

A fase pode assumir um dos seguintes valores:

- `Bound`: O `TridentBackendConfig` CR está associado a um backend, e esse backend contém `configRef` definido para o `TridentBackendConfig` uid do CR.
- `Unbound`: Representado usando `""`. O `TridentBackendConfig` objeto não está vinculado a um backend. Todos os CRs recém-criados `TridentBackendConfig` estão nesta fase por padrão. Após a mudança de fase, ele não pode retornar ao estado Unbound novamente.
- `Deleting`: O `TridentBackendConfig` CR `deletionPolicy` foi configurado para ser excluído. Quando o `TridentBackendConfig` CR é excluído, ele transita para o estado Deleting.
  - Caso não existam reivindicações de volume persistentes (PVCs) no backend, excluir o `TridentBackendConfig` resultará na exclusão do backend pelo Trident, bem como da `TridentBackendConfig` CR.
  - Se um ou mais PVCs estiverem presentes no backend, ele entra em estado de exclusão. O `TridentBackendConfig` CR subsequentemente também entra na fase de exclusão. O backend e `TridentBackendConfig` são excluídos somente após todos os PVCs serem excluídos.
- `Lost`: O backend associado ao `TridentBackendConfig` CR foi excluído acidentalmente ou deliberadamente e o `TridentBackendConfig` CR ainda possui uma referência ao backend excluído. O `TridentBackendConfig` CR ainda pode ser excluído independentemente do `deletionPolicy` valor.
- `Unknown`: Trident não consegue determinar o estado ou a existência do backend associado ao `TridentBackendConfig` CR. Por exemplo, se o servidor da API não estiver respondendo ou se o `tridentbackends.trident.netapp.io` CRD estiver ausente. Isso pode exigir intervenção.

Nesta etapa, um backend foi criado com sucesso! Existem diversas operações que também podem ser realizadas, como "atualizações de backend e exclusões de backend".

(Opcional) Passo 4: obtenha mais detalhes

Você pode executar o seguinte comando para obter mais informações sobre seu backend:

```
kubectl -n trident get tbc backend-tbc-ontap-san -o wide
```

```
NAME                     BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   8d24fce7-6f60-4d4a-8ef6-
bab2699e6ab8   Bound   Success   ontap-san        delete
```

Além disso, você também pode obter um dump YAML/JSON de `TridentBackendConfig`.

```
kubectl -n trident get tbc backend-tbc-ontap-san -o yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  creationTimestamp: 2021-04-21T20:45:11Z
  finalizers:
    - trident.netapp.io
  generation: 1
  name: backend-tbc-ontap-san
  namespace: trident
  resourceVersion: "947143"
  uid: 35b9d777-109f-43d5-8077-c74a4559d09c
spec:
  backendName: ontap-san-backend
  credentials:
    name: backend-tbc-ontap-san-secret
  managementLIF: 10.0.0.1
  dataLIF: 10.0.0.2
  storageDriverName: ontap-san
  svm: trident_svm
  version: 1
status:
  backendInfo:
    backendName: ontap-san-backend
    backendUUID: 8d24fce7-6f60-4d4a-8ef6-bab2699e6ab8
  deletionPolicy: delete
  lastOperationStatus: Success
  message: Backend 'ontap-san-backend' created
  phase: Bound
```

`backendInfo` contém o `backendName` e o `backendUUID` do backend criado em resposta ao `TridentBackendConfig` CR. O `lastOperationStatus` campo representa o status da última operação do `TridentBackendConfig` CR, que pode ser iniciada pelo usuário (por exemplo, o usuário alterou algo em `spec`) ou iniciada pelo Trident (por exemplo, durante reinicializações do Trident). Pode ser Success ou Failed. `phase` representa o status da relação entre o `TridentBackendConfig` CR e o backend. No exemplo acima, `phase` tem o valor Bound, o que significa que o `TridentBackendConfig` CR está associado ao backend.

Você pode executar o `kubectl -n trident describe tbc <tbc-cr-name>` comando para obter detalhes dos registros de eventos.

Não é possível atualizar ou excluir um backend que contenha um objeto associado `TridentBackendConfig` usando `tridentctl`. Para entender as etapas envolvidas na troca entre `tridentctl` e `TridentBackendConfig`, "veja aqui".

Gerenciar backends

Realize o gerenciamento de backend com kubectl

Aprenda sobre como realizar operações de gerenciamento de backend usando `kubectl`.

Excluir um backend

Ao excluir um `TridentBackendConfig`, você instrui Trident a excluir/manter backends (com base em `deletionPolicy`). Para excluir um backend, certifique-se de que `deletionPolicy` esteja definido como delete. Para excluir apenas o `TridentBackendConfig`, certifique-se de que `deletionPolicy` esteja definido como retain. Isso garante que o backend ainda esteja presente e possa ser gerenciado usando `tridentctl`.

Execute o seguinte comando:

```
kubectl delete tbc <tbc-name> -n trident
```

Trident não exclui os segredos do Kubernetes que estavam em uso por `TridentBackendConfig`. O usuário do Kubernetes é responsável por limpar os segredos. É preciso ter cuidado ao excluir segredos. Você só deve excluir segredos se eles não estiverem em uso pelos backends.

Veja os backends existentes

Execute o seguinte comando:

```
kubectl get tbc -n trident
```

Você também pode executar `tridentctl get backend -n trident` ou `tridentctl get backend -o yaml -n trident` para obter uma lista de todos os backends existentes. Essa lista também incluirá backends que foram criados com `tridentctl`.

Atualizar um backend

Podem haver vários motivos para atualizar um backend:

- As credenciais do sistema de storage foram alteradas. Para atualizar as credenciais, o Kubernetes Secret usado no `TridentBackendConfig` objeto deve ser atualizado. Trident atualizará automaticamente o backend com as credenciais mais recentes fornecidas. Execute o seguinte comando para atualizar o Kubernetes Secret:

  ```
  kubectl apply -f <updated-secret-file.yaml> -n trident
  ```

- Parâmetros (como o nome da SVM do ONTAP que está sendo usada) precisam ser atualizados.
  - Você pode atualizar `TridentBackendConfig` objetos diretamente pelo Kubernetes usando o seguinte comando:

```
kubectl apply -f <updated-backend-file.yaml>
```

  - Alternativamente, você pode fazer alterações no CR existente `TridentBackendConfig` usando o seguinte comando:

```
kubectl edit tbc <tbc-name> -n trident
```

- Se uma atualização do backend falhar, o backend continuará em sua última configuração conhecida. Você pode visualizar os logs para determinar a causa executando `kubectl get tbc <tbc-name> -o yaml -n trident` ou `kubectl describe tbc <tbc-name> -n trident`.
- Após identificar e corrigir o problema com o arquivo de configuração, você pode executar novamente o comando de atualização.

Realize o gerenciamento de backend com tridentctl

Saiba como realizar operações de gerenciamento de backend usando `tridentctl`.

Criar um backend

Após criar um "arquivo de configuração backend", execute o seguinte comando:

```
tridentctl create backend -f <backend-file> -n trident
```

Se a criação do backend falhar, algo estava errado com a configuração do backend. Você pode visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o comando `create` novamente.

Excluir um backend

Para excluir um backend do Trident, faça o seguinte:

1. Recupere o nome do backend:

```
tridentctl get backend -n trident
```

2. Exclua o backend:

```
tridentctl delete backend <backend-name> -n trident
```

Se Trident tiver provisionado volumes e snapshots desse backend que ainda existam, a exclusão do backend impede que novos volumes sejam provisionados por ele. O backend continuará existindo no estado "Deleting".

Veja os backends existentes

Para visualizar os backends que Trident conhece, faça o seguinte:

- Para obter um resumo, execute o seguinte comando:

```
tridentctl get backend -n trident
```

- Para obter todos os detalhes, execute o seguinte comando:

```
tridentctl get backend -o json -n trident
```

Atualizar um backend

Após criar um novo arquivo de configuração backend, execute o seguinte comando:

```
tridentctl update backend <backend-name> -f <backend-file> -n trident
```

Se a atualização do backend falhar, algo estava errado com a configuração do backend ou você tentou uma atualização inválida. Você pode visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs -n trident
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode simplesmente executar o `update` comando novamente.

Identifique as classes de armazenamento que usam um backend

Este é um exemplo do tipo de perguntas que você pode responder com o JSON que `tridentctl` gera para objetos do backend. Isso usa o utilitário `jq`, que você precisa instalar.

```
tridentctl get backend -o json | jq '[.items[] | {backend: .name,
storageClasses: [.storage[].storageClasses]|unique}]'
```

Isso também se aplica a backends que foram criados usando `TridentBackendConfig`.

Alternar entre opções de gerenciamento de backend

Aprenda sobre as diferentes maneiras de gerenciar backends no Trident.

Opções para gerenciamento de backends

Com a introdução de `TridentBackendConfig`, os administradores agora têm duas maneiras únicas de gerenciar backends. Isso levanta as seguintes questões:

- Backends criados usando `tridentctl` podem ser gerenciados com `TridentBackendConfig`?
- Backends criados usando `TridentBackendConfig` podem ser gerenciados usando `tridentctl`?

Gerencie `tridentctl` back-ends usando `TridentBackendConfig`

Esta seção aborda os passos necessários para gerenciar backends que foram criados usando `tridentctl` diretamente através da interface do Kubernetes, criando `TridentBackendConfig` objetos.

Isso se aplica aos seguintes cenários:

- Backends pré-existentes, que não possuem um `TridentBackendConfig` porque foram criados com `tridentctl`.
- Novos backends que foram criados com `tridentctl`, enquanto outros `TridentBackendConfig` objetos existem.

Em ambos os cenários, os backends continuarão presentes, com Trident agendando volumes e operando sobre eles. Os administradores têm uma das duas opções aqui:

- Continue usando `tridentctl` para gerenciar os backends que foram criados com ele.
- Vincule os backends criados usando `tridentctl` a um novo `TridentBackendConfig` objeto. Fazer isso significa que os backends serão gerenciados usando `kubectl` e não `tridentctl`.

Para gerenciar um backend preexistente usando `kubectl`, você precisará criar um `TridentBackendConfig` que se conecte ao backend existente. Aqui está uma visão geral de como isso funciona:

1. Crie um segredo do Kubernetes. O segredo contém as credenciais que Trident precisa para se comunicar com o cluster/serviço de armazenamento.
2. Crie um `TridentBackendConfig` objeto. Isso contém detalhes sobre o cluster/serviço de storage e faz referência ao segredo criado na etapa anterior. É preciso ter cuidado para especificar parâmetros de configuração idênticos (como `spec.backendName`, `spec.storagePrefix`, `spec.storageDriverName`, e assim por diante). `spec.backendName` deve ser definido com o nome do backend existente.

Etapa 0: identificar o backend

Para criar uma `TridentBackendConfig` que se vincula a um backend existente, você precisará obter a configuração do backend. Neste exemplo, vamos supor que um backend foi criado usando a seguinte definição JSON:

```
tridentctl get backend ontap-nas-backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

```
cat ontap-nas-backend.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.10.10.1",
  "dataLIF": "10.10.10.2",
  "backendName": "ontap-nas-backend",
  "svm": "trident_svm",
  "username": "cluster-admin",
  "password": "admin-password",
  "defaults": {
    "spaceReserve": "none",
    "encryption": "false"
  },
  "labels": {
    "store": "nas_store"
  },
  "region": "us_east_1",
  "storage": [
    {
      "labels": {
        "app": "msoffice",
        "cost": "100"
      },
      "zone": "us_east_1a",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "true",
        "unixPermissions": "0755"
      }
    },
    {
      "labels": {
        "app": "mysqldb",
        "cost": "25"
      },
      "zone": "us_east_1d",
      "defaults": {
        "spaceReserve": "volume",
        "encryption": "false",
        "unixPermissions": "0775"
      }
    }
  ]
}
```

Passo 1: criar um segredo do Kubernetes

Crie um segredo que contenha as credenciais para o backend, conforme mostrado neste exemplo:

```
cat tbc-ontap-nas-backend-secret.yaml
```

```
apiVersion: v1
kind: Secret
metadata:
  name: ontap-nas-backend-secret
type: Opaque
stringData:
  username: cluster-admin
  password: admin-password
```

```
kubectl create -f tbc-ontap-nas-backend-secret.yaml -n trident
secret/backend-tbc-ontap-san-secret created
```

Etapa 2: Criar um `TridentBackendConfig` CR

O próximo passo é criar uma `TridentBackendConfig` CR que se vinculará automaticamente à já existente `ontap-nas-backend` (como neste exemplo). Certifique-se de que os seguintes requisitos sejam atendidos:

- O mesmo nome de backend está definido em `spec.backendName`.
- Os parâmetros de configuração são idênticos aos do backend original.
- Pools virtuais (se presentes) devem manter a mesma ordem que no backend original.
- As credenciais são fornecidas por meio de um Kubernetes Secret e não em texto simples.

Nesse caso, o `TridentBackendConfig` ficará assim:

```
cat backend-tbc-ontap-nas.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: tbc-ontap-nas-backend
spec:
  version: 1
  storageDriverName: ontap-nas
  managementLIF: 10.10.10.1
  dataLIF: 10.10.10.2
  backendName: ontap-nas-backend
  svm: trident_svm
  credentials:
    name: mysecret
  defaults:
    spaceReserve: none
    encryption: 'false'
  labels:
    store: nas_store
  region: us_east_1
  storage:
  - labels:
      app: msoffice
      cost: '100'
    zone: us_east_1a
    defaults:
      spaceReserve: volume
      encryption: 'true'
      unixPermissions: '0755'
  - labels:
      app: mysqldb
      cost: '25'
    zone: us_east_1d
    defaults:
      spaceReserve: volume
      encryption: 'false'
      unixPermissions: '0775'
```

```
kubectl create -f backend-tbc-ontap-nas.yaml -n trident
tridentbackendconfig.trident.netapp.io/tbc-ontap-nas-backend created
```

Etapa 3: Verificar o status do `TridentBackendConfig`CR

Após o `TridentBackendConfig` ter sido criado, sua fase deve ser `Bound`. Ele também deve refletir o mesmo nome de backend e UUID do backend existente.

```
kubectl get tbc tbc-ontap-nas-backend -n trident
NAME                   BACKEND NAME          BACKEND UUID
PHASE   STATUS
tbc-ontap-nas-backend  ontap-nas-backend     52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7   Bound   Success

#confirm that no new backends were created (i.e., TridentBackendConfig did
not end up creating a new backend)
tridentctl get backend -n trident
+---------------------+----------------
+--------------------------------------+--------+---------+
|          NAME       | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+---------------------+----------------
+--------------------------------------+--------+---------+
| ontap-nas-backend   | ontap-nas      | 52f2eb10-e4c6-4160-99fc-
96b3be5ab5d7 | online |      25 |
+---------------------+----------------
+--------------------------------------+--------+---------+
```

O backend será agora completamente gerenciado usando o `tbc-ontap-nas-backend TridentBackendConfig` objeto.

Gerencie `TridentBackendConfig` back-ends usando `tridentctl`

```
`tridentctl` pode ser usado para listar backends que foram criados usando
`TridentBackendConfig`. Além disso, administradores também podem optar por
gerenciar completamente esses backends através de  `tridentctl`, excluindo
`TridentBackendConfig` e certificando-se de que  `spec.deletionPolicy`
esteja definido como  `retain`.
```

Etapa 0: identificar o backend

Por exemplo, vamos supor que o seguinte backend foi criado usando `TridentBackendConfig`:

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

A partir da saída, observa-se que `TridentBackendConfig` foi criado com sucesso e está vinculado a um backend [observe o UUID do backend].

Etapa 1: confirmar `deletionPolicy` **está definido como** `retain`

Vamos analisar o valor de `deletionPolicy`. Isso precisa ser definido como `retain`. Isso garante que, quando um `TridentBackendConfig` CR for excluído, a definição de backend ainda estará presente e poderá ser gerenciada com `tridentctl`.

```
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        delete

# Patch value of deletionPolicy to retain
kubectl patch tbc backend-tbc-ontap-san --type=merge -p
'{"spec":{"deletionPolicy":"retain"}}' -n trident
tridentbackendconfig.trident.netapp.io/backend-tbc-ontap-san patched

#Confirm the value of deletionPolicy
kubectl get tbc backend-tbc-ontap-san -n trident -o wide
NAME                    BACKEND NAME        BACKEND UUID
PHASE   STATUS    STORAGE DRIVER   DELETION POLICY
backend-tbc-ontap-san   ontap-san-backend   81abcb27-ea63-49bb-b606-
0a5315ac5f82   Bound   Success   ontap-san        retain
```

Não prossiga para a próxima etapa a menos que `deletionPolicy` esteja definido como `retain`.

Passo 2: Exclua o `TridentBackendConfig` CR

A etapa final é excluir o `TridentBackendConfig` CR. Após confirmar que o `deletionPolicy` está definido como `retain`, você pode prosseguir com a exclusão:

```
kubectl delete tbc backend-tbc-ontap-san -n trident
tridentbackendconfig.trident.netapp.io "backend-tbc-ontap-san" deleted

tridentctl get backend ontap-san-backend -n trident
+-------------------+----------------
+--------------------------------------+--------+---------+
|       NAME        | STORAGE DRIVER |                 UUID
| STATE  | VOLUMES |
+-------------------+----------------
+--------------------------------------+--------+---------+
| ontap-san-backend | ontap-san      | 81abcb27-ea63-49bb-b606-
0a5315ac5f82 | online |      33 |
+-------------------+----------------
+--------------------------------------+--------+---------+
```

Ao excluir o `TridentBackendConfig` objeto, Trident simplesmente o remove sem realmente excluir o próprio backend.

Criar e gerenciar classes de armazenamento

Crie uma storage class

Configure um objeto StorageClass do Kubernetes e crie a storage class para instruir Trident sobre como provisionar volumes.

Configure um objeto StorageClass do Kubernetes

O "Objeto Kubernetes StorageClass"identifica Trident como o provisionador usado para essa classe e instrui Trident sobre como provisionar um volume. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-gold
provisioner: csi.trident.netapp.io
mountOptions:
  - nfsvers=3
  - nolock
parameters:
  backendType: "ontap-nas"
  media: "ssd"
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Crie uma storage class

Após criar o StorageClass objeto, você pode criar a classe de armazenamento. Exemplos de storage class fornece alguns exemplos básicos que você pode usar ou modificar.

Passos

1. Este é um objeto do Kubernetes, portanto, use `kubectl` para criá-lo no Kubernetes.

```
kubectl create -f sample-input/storage-class-basic-csi.yaml
```

2. Agora você deverá ver uma classe de storage **basic-csi** tanto no Kubernetes quanto no Trident, e o Trident deverá ter descoberto os pools no backend.

```
kubectl get sc basic-csi
```

```
NAME         PROVISIONER             AGE
basic-csi    csi.trident.netapp.io   15h
```

```
./tridentctl -n trident get storageclass basic-csi -o json
```

```
{
  "items": [
    {
      "Config": {
        "version": "1",
        "name": "basic-csi",
        "attributes": {
          "backendType": "ontap-nas"
        },
        "storagePools": null,
        "additionalStoragePools": null
      },
      "storage": {
        "ontapnas_10.0.0.1": [
          "aggr1",
          "aggr2",
          "aggr3",
          "aggr4"
        ]
      }
    }
  ]
}
```


Exemplos de storage class

Trident fornece "Definições simples de storage class para backends específicos".

Alternativamente, você pode editar `sample-input/storage-class-csi.yaml.templ` o arquivo que acompanha o instalador e substituir *`BACKEND_TYPE`* pelo nome do driver de storage.

```
./tridentctl -n trident get backend
+-------------+---------------+--------------------------------------
+--------+---------+
|    NAME     | STORAGE DRIVER |                 UUID                 |
STATE  | VOLUMES |
+-------------+---------------+--------------------------------------
+--------+---------+
| nas-backend | ontap-nas      | 98e19b74-aec7-4a3d-8dcf-128e5033b214 |
online |       0 |
+-------------+---------------+--------------------------------------
+--------+---------+

cp sample-input/storage-class-csi.yaml.templ sample-input/storage-class-
basic-csi.yaml

# Modify __BACKEND_TYPE__ with the storage driver field above (e.g.,
ontap-nas)
vi sample-input/storage-class-basic-csi.yaml
```

Gerenciar classes de armazenamento

Você pode visualizar as classes de armazenamento existentes, definir uma classe de armazenamento padrão, identificar o backend da classe de armazenamento e excluir classes de armazenamento.

Veja as classes de armazenamento existentes

- Para visualizar as classes de armazenamento do Kubernetes existentes, execute o seguinte comando:

```
kubectl get storageclass
```

- Para visualizar os detalhes da storage class do Kubernetes, execute o seguinte comando:

```
kubectl get storageclass <storage-class> -o json
```

- Para visualizar as classes de armazenamento sincronizadas do Trident, execute o seguinte comando:

```
tridentctl get storageclass
```

- Para visualizar os detalhes da classe de armazenamento sincronizado do Trident, execute o seguinte comando:

```
tridentctl get storageclass <storage-class> -o json
```

Defina uma classe de armazenamento padrão

O Kubernetes 1.6 adicionou a capacidade de definir uma classe de armazenamento padrão. Essa é a classe de armazenamento que será usada para provisionar um Volume Persistente caso o usuário não especifique uma em uma Solicitação de Volume Persistente (PVC).

- Defina uma classe de armazenamento padrão configurando a anotação `storageclass.kubernetes.io/is-default-class` como verdadeira na definição da classe de armazenamento. De acordo com a especificação, qualquer outro valor ou a ausência da anotação é interpretado como falso.
- Você pode configurar uma classe de armazenamento existente para ser a classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"true"}}}'
```

- Da mesma forma, você pode remover a anotação da classe de armazenamento padrão usando o seguinte comando:

```
kubectl patch storageclass <storage-class-name> -p '{"metadata":
{"annotations":{"storageclass.kubernetes.io/is-default-class":"false"}}}'
```

Existem também exemplos no pacote de instalação do Trident que incluem essa anotação.

> Deve haver apenas uma classe de armazenamento padrão em seu cluster por vez. O Kubernetes não impede tecnicamente que você tenha mais de uma, mas se comportará como se não houvesse classe de armazenamento padrão.

Identifique o backend para uma storage class

Este é um exemplo do tipo de pergunta que você pode responder com o JSON que `tridentctl` gera para objetos de backend do Trident. Isso utiliza o utilitário `jq`, que talvez você precise instalar primeiro.

```
tridentctl get storageclass -o json | jq  '[.items[] | {storageClass:
.Config.name, backends: [.storage]|unique}]'
```

Excluir uma storage class

Para excluir uma classe de armazenamento do Kubernetes, execute o seguinte comando:

```
kubectl delete storageclass <storage-class>
```

`<storage-class>` deve ser substituído pela sua classe de armazenamento.

Quaisquer volumes persistentes que foram criados por meio dessa classe de armazenamento permanecerão intactos, e Trident continuará a gerenciá-los.

> Trident impõe um espaço em branco `fsType` para os volumes que cria. Para backends iSCSI, recomenda-se impor `parameters.fsType` no StorageClass. Você deve excluir as StorageClasses existentes e recriá-las com `parameters.fsType` especificado.

Provisionar e gerenciar volumes

Provisionar um volume

Crie um PersistentVolumeClaim (PVC) que utiliza o StorageClass do Kubernetes configurado para solicitar acesso ao PV. Em seguida, você pode montar o PV em um pod.

Visão geral

Um "*PersistentVolumeClaim*" (PVC) é uma solicitação de acesso ao PersistentVolume no cluster.

O PVC pode ser configurado para solicitar armazenamento de um determinado tamanho ou modo de acesso. Usando o StorageClass associado, o administrador do cluster pode controlar mais do que apenas o tamanho e o modo de acesso do PersistentVolume—como desempenho ou nível de serviço.

Após criar o PVC, você pode montar o volume em um pod.

Crie o PVC

Passos

1. Crie o PVC.

```
kubectl create -f pvc.yaml
```

2. Verifique o status do PVC.

```
kubectl get pvc
```

```
NAME          STATUS   VOLUME     CAPACITY   ACCESS MODES   STORAGECLASS AGE
pvc-storage   Bound    pv-name    1Gi        RWO                    5m
```

1. Monte o volume em um pod.

```
kubectl create -f pv-pod.yaml
```

Você pode monitorar o progresso usando `kubectl get pod --watch`.

2. Verifique se o volume está montado em `/my/mount/path`.

```
kubectl exec -it task-pv-pod -- df -h /my/mount/path
```

3. Agora você pode excluir o Pod. O aplicativo Pod deixará de existir, mas o volume permanecerá.

```
kubectl delete pod pv-pod
```

Exemplos de manifestos

Manifestos de exemplo de PersistentVolumeClaim

Estes exemplos mostram opções básicas de configuração de PVC.

PVC com acesso RWO

Este exemplo mostra um PVC básico com acesso RWO associado a um StorageClass chamado `basic-csi`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc-storage
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: basic-csi
```

PVC com NVMe/TCP

Este exemplo mostra um PVC básico para NVMe/TCP com acesso RWO que está associado a um StorageClass chamado `protection-gold`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
name: pvc-san-nvme
spec:
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: protection-gold
```

Exemplos de manifesto de pod

Estes exemplos mostram configurações básicas para anexar o PVC a um pod.

Configuração básica

```
kind: Pod
apiVersion: v1
metadata:
  name: pv-pod
spec:
  volumes:
    - name: storage
      persistentVolumeClaim:
       claimName: pvc-storage
  containers:
    - name: pv-container
      image: nginx
      ports:
        - containerPort: 80
          name: "http-server"
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: storage
```

Configuração básica de NVMe/TCP

```
apiVersion: v1
kind: Pod
metadata:
  name: pod-nginx
spec:
  volumes:
    - name: basic-pvc
      persistentVolumeClaim:
        claimName: pvc-san-nvme
  containers:
    - name: task-pv-container
      image: nginx
      volumeMounts:
        - mountPath: "/my/mount/path"
          name: basic-pvc
```

Consulte "Objetos Kubernetes e Trident" para obter detalhes sobre como as classes de armazenamento interagem com o `PersistentVolumeClaim` e os parâmetros para controlar como Trident provisiona volumes.

Expandir volumes

Trident oferece aos usuários do Kubernetes a capacidade de expandir seus volumes após eles serem criados. Encontre informações sobre as configurações necessárias para expandir volumes iSCSI, NFS, SMB, NVMe/TCP e FC.

Expandir um volume iSCSI

Você pode expandir um iSCSI Persistent Volume (PV) usando o provisionador CSI.

> A expansão de volume iSCSI é suportada pelos `ontap-san`, `ontap-san-economy`, `solidfire-san` drivers e requer Kubernetes 1.16 ou posterior.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Edite a definição de StorageClass para definir o campo `allowVolumeExpansion` como `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o novo tamanho desejado, que deve ser maior que o tamanho original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Etapa 3: defina um pod que conecte o PVC

Conecte o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um PV iSCSI:

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da operação de expansão.

Neste exemplo, um pod é criado que usa o `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Etapa 4: expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do PVC e atualize o `spec.resources.requests.storage` para 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Etapa 5: valide a expansão

Você pode validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do volume do Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Expandir um volume FC

Você pode expandir um FC Persistent Volume (PV) usando o provisionador CSI.

> A expansão de volume FC é suportada pelo `ontap-san` driver e requer Kubernetes 1.16 ou posterior.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Edite a definição de StorageClass para definir o campo `allowVolumeExpansion` como `true`.

```
cat storageclass-ontapsan.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-san
provisioner: csi.trident.netapp.io
parameters:
  backendType: "ontap-san"
allowVolumeExpansion: True
```

Para um StorageClass já existente, edite-o para incluir o `allowVolumeExpansion` parâmetro.

Passo 2: Crie um PVC com o StorageClass que você criou

Edite a definição de PVC e atualize o `spec.resources.requests.storage` para refletir o novo tamanho desejado, que deve ser maior que o tamanho original.

```
cat pvc-ontapsan.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: san-pvc
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: ontap-san
```

Trident cria um Persistent Volume (PV) e o associa a este Persistent Volume Claim (PVC).

```
kubectl get pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi
RWO            ontap-san      8s

kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   1Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               10s
```

Etapa 3: defina um pod que conecte o PVC

Conecte o PV a um pod para que ele seja redimensionado. Existem dois cenários ao redimensionar um PV FC:

- Se o PV estiver conectado a um pod, Trident expande o volume no backend de armazenamento, faz uma nova varredura no dispositivo e redimensiona o sistema de arquivos.
- Ao tentar redimensionar um PV não anexado, Trident expande o volume no backend de armazenamento. Depois que o PVC é vinculado a um pod, Trident verifica novamente o dispositivo e redimensiona o sistema de arquivos. O Kubernetes atualiza o tamanho do PVC após a conclusão bem-sucedida da

operação de expansão.

Neste exemplo, um pod é criado que usa o `san-pvc`.

```
 kubectl get pod
NAME         READY   STATUS    RESTARTS   AGE
ubuntu-pod   1/1     Running   0          65s

 kubectl describe pvc san-pvc
Name:          san-pvc
Namespace:     default
StorageClass:  ontap-san
Status:        Bound
Volume:        pvc-8a814d62-bd58-4253-b0d1-82f2885db671
Labels:        <none>
Annotations:   pv.kubernetes.io/bind-completed: yes
               pv.kubernetes.io/bound-by-controller: yes
               volume.beta.kubernetes.io/storage-provisioner:
csi.trident.netapp.io
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:      1Gi
Access Modes:  RWO
VolumeMode:    Filesystem
Mounted By:    ubuntu-pod
```

Etapa 4: expandir o PV

Para redimensionar o PV criado de 1Gi para 2Gi, edite a definição do PVC e atualize o `spec.resources.requests.storage` para 2Gi.

```
kubectl edit pvc san-pvc
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: "2019-10-10T17:32:29Z"
  finalizers:
  - kubernetes.io/pvc-protection
  name: san-pvc
  namespace: default
  resourceVersion: "16609"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/san-pvc
  uid: 8a814d62-bd58-4253-b0d1-82f2885db671
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 2Gi
 # ...
```

Etapa 5: valide a expansão

Você pode validar se a expansão funcionou corretamente conferindo o tamanho do PVC, PV e do volume do Trident:

```
kubectl get pvc san-pvc
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS   AGE
san-pvc   Bound    pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi
RWO            ontap-san      11m
kubectl get pv
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM              STORAGECLASS   REASON   AGE
pvc-8a814d62-bd58-4253-b0d1-82f2885db671   2Gi        RWO
Delete           Bound    default/san-pvc    ontap-san               12m
tridentctl get volumes -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-8a814d62-bd58-4253-b0d1-82f2885db671 | 2.0 GiB | ontap-san     |
block    | a9b7bfff-0505-4e31-b6c5-59f492e02d33 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Expandir um volume NFS

Trident suporta expansão de volume para NFS PVs provisionados em `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, e `azure-netapp-files` backends.

Passo 1: Configurar o StorageClass para suportar a expansão de volume

Para redimensionar um NFS PV, o administrador primeiro precisa configurar a classe de armazenamento para permitir expansão de volume, definindo o campo `allowVolumeExpansion` como `true`:

```
cat storageclass-ontapnas.yaml
```

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontapnas
provisioner: csi.trident.netapp.io
parameters:
  backendType: ontap-nas
allowVolumeExpansion: true
```

Se você já criou uma classe de armazenamento sem essa opção, basta editar a classe de armazenamento

existente usando `kubectl edit storageclass` para permitir a expansão de volume.

Passo 2: Crie um PVC com o StorageClass que você criou

```
cat pvc-ontapnas.yaml
```

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: ontapnas20mb
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 20Mi
  storageClassName: ontapnas
```

Trident deve criar um NFS PV de 20 MiB para este PVC:

```
kubectl get pvc
NAME           STATUS    VOLUME
CAPACITY     ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi
RWO            ontapnas        9s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                   STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   20Mi       RWO
Delete           Bound    default/ontapnas20mb   ontapnas
2m42s
```

Etapa 3: expandir o PV

Para redimensionar o PV recém-criado de 20 MiB para 1 GiB, edite o PVC e defina `spec.resources.requests.storage` para 1 GiB:

```
kubectl edit pvc ontapnas20mb
```

```
# Please edit the object below. Lines beginning with a '#' will be
ignored,
# and an empty file will abort the edit. If an error occurs while saving
this file will be
# reopened with the relevant failures.
#
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  annotations:
    pv.kubernetes.io/bind-completed: "yes"
    pv.kubernetes.io/bound-by-controller: "yes"
    volume.beta.kubernetes.io/storage-provisioner: csi.trident.netapp.io
  creationTimestamp: 2018-08-21T18:26:44Z
  finalizers:
  - kubernetes.io/pvc-protection
  name: ontapnas20mb
  namespace: default
  resourceVersion: "1958015"
  selfLink: /api/v1/namespaces/default/persistentvolumeclaims/ontapnas20mb
  uid: c1bd7fa5-a56f-11e8-b8d7-fa163e59eaab
spec:
  accessModes:
  - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
# ...
```

Etapa 4: valide a expansão

Você pode verificar se o redimensionamento funcionou corretamente conferindo o tamanho do PVC, do PV e do Trident volume:

```
kubectl get pvc ontapnas20mb
NAME           STATUS   VOLUME
CAPACITY   ACCESS MODES   STORAGECLASS    AGE
ontapnas20mb   Bound    pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi
RWO            ontapnas        4m44s

kubectl get pv pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7
NAME                                       CAPACITY   ACCESS MODES
RECLAIM POLICY   STATUS   CLAIM                  STORAGECLASS    REASON
AGE
pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7   1Gi        RWO
Delete           Bound    default/ontapnas20mb   ontapnas
5m35s

tridentctl get volume pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 -n trident
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
| pvc-08f3d561-b199-11e9-8d9f-5254004dfdb7 | 1.0 GiB | ontapnas      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+-----------------------------------------+---------+---------------
+----------+-----------------------------------------+--------+---------+
```

Importar volumes

Você pode importar volumes de armazenamento existentes como um PV do Kubernetes usando `tridentctl import` ou criando uma Persistent Volume Claim (PVC) com anotações de importação do Trident.

Visão geral e considerações

Você pode importar um volume para Trident para:

- Containerize uma aplicação e reutilize seu conjunto de dados existente
- Utilize um clone de um conjunto de dados para uma aplicação efêmera
- Reconstruir um cluster Kubernetes com falha
- Migrar dados de aplicativo durante a recuperação de desastres

Considerações

Antes de importar um volume, revise as seguintes considerações.

- Trident pode importar apenas volumes ONTAP do tipo RW (leitura e gravação). Volumes do tipo DP (proteção de dados) são volumes de destino do SnapMirror. Você deve quebrar a relação de

espelhamento antes de importar o volume para Trident.

- Sugerimos importar volumes sem conexões ativas. Para importar um volume em uso ativo, clone o volume e depois execute a importação.

> Isso é especialmente importante para volumes de bloco, pois o Kubernetes desconheceria a conexão anterior e poderia facilmente anexar um volume ativo a um pod. Isso pode resultar em corrupção de dados.

- Embora `StorageClass` deva ser especificado em um PVC, Trident não utiliza esse parâmetro durante a importação. As classes de armazenamento são usadas durante a criação do volume para selecionar entre os pools disponíveis com base nas características de armazenamento. Como o volume já existe, nenhuma seleção de pool é necessária durante a importação. Portanto, a importação não falhará mesmo que o volume exista em um backend ou pool que não corresponda à classe de armazenamento especificada no PVC.
- O tamanho do volume existente é determinado e definido no PVC. Após o volume ser importado pelo driver de armazenamento, o PV é criado com um ClaimRef para o PVC.
  - A política de recuperação é inicialmente definida como `retain` no PV. Depois que o Kubernetes vincula com sucesso o PVC e o PV, a política de recuperação é atualizada para corresponder à política de recuperação da Storage Class.
  - Se a política de recuperação da Storage Class for `delete`, o volume de armazenamento será excluído quando o PV for excluído.
- Por padrão, Trident gerencia o PVC e renomeia o FlexVol volume e o LUN no backend. Você pode usar a `--no-manage` flag para importar um volume não gerenciado e a `--no-rename` flag para manter o nome do volume.
  - `--no-manage`* - Se você usar a `--no-manage` flag, Trident não executa nenhuma operação adicional no PVC ou PV para o ciclo de vida dos objetos. O volume de armazenamento não é excluído quando o PV é excluído e outras operações, como clone de volume e expansão de volume, também são ignoradas.
  - `--no-rename`* - Se você usar a `--no-rename`flag, Trident mantém o nome do volume existente ao importar volumes e gerencia o ciclo de vida dos volumes. Essa opção é compatível apenas para os drivers `ontap-nas`, `ontap-san` (incluindo sistemas ASA r2) e `ontap-san-economy`.

> Essas opções são úteis se você deseja usar o Kubernetes para cargas de trabalho em contêineres, mas de outra forma deseja gerenciar o ciclo de vida do volume de armazenamento fora do Kubernetes.

- Uma anotação é adicionada ao PVC e ao PV com a dupla função de indicar que o volume foi importado e se o PVC e o PV são gerenciados. Essa anotação não deve ser modificada ou removida.

Importar um volume

Você pode importar um volume usando `tridentctl import` ou criando um PVC com anotações de importação do Trident.

> Se você utiliza anotações PVC, não precisa baixar ou usar `tridentctl` para importar o volume.

Usando tridentctl

Passos

1. Crie um arquivo PVC (por exemplo, `pvc.yaml`) que será usado para criar o PVC. O arquivo PVC deve incluir `name`, `namespace`, `accessModes` e `storageClassName`. Opcionalmente, você pode especificar `unixPermissions` na sua definição de PVC.

   A seguir está um exemplo de especificação mínima:

   ```
   kind: PersistentVolumeClaim
   apiVersion: v1
   metadata:
     name: my_claim
     namespace: my_namespace
   spec:
     accessModes:
       - ReadWriteOnce
     storageClassName: my_storage_class
   ```

   Inclua apenas os parâmetros necessários. Parâmetros adicionais, como o nome do PV ou o tamanho do volume, podem fazer com que o comando de importação falhe.

2. Use o `tridentctl import` comando para especificar o nome do backend Trident que contém o volume e o nome que identifica exclusivamente o volume no armazenamento (por exemplo: ONTAP FlexVol, Element Volume). O `-f` argumento é obrigatório para especificar o caminho para o arquivo PVC.

   ```
   tridentctl import volume <backendName> <volumeName> -f <path-to-pvc-file>
   ```

Usando anotações em PVC

Passos

1. Crie um arquivo YAML de PVC (por exemplo, `pvc.yaml`) com as anotações de importação do Trident necessárias. O arquivo de PVC deve incluir:

   - `name` e `namespace` em metadados
   - `accessModes`, `resources.requests.storage`, e `storageClassName` em especificação
   - Anotações:
     - `trident.netapp.io/importOriginalName`: Nome do volume no backend
     - `trident.netapp.io/importBackendUUID`: UUID do backend onde o volume existe
     - `trident.netapp.io/notManaged` (*Opcional*): Defina como `"true"` para volumes não gerenciados. O padrão é `"false"`.

       A seguir está um exemplo de especificação para importar um volume gerenciado:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <pvc-name>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "<volume-name>"
    trident.netapp.io/importBackendUUID: "<backend-uuid>"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: <size>
  storageClassName: <storage-class-name>
```

2. Aplique o arquivo YAML do PVC ao seu cluster Kubernetes:

```
kubectl apply -f <pvc-file>.yaml
```

Trident importará automaticamente o volume e o vinculará ao PVC.

Exemplos

Revise os seguintes exemplos de importação de volume para drivers compatíveis.

ONTAP NAS e ONTAP NAS FlexGroup

Trident suporta importação de volume usando os `ontap-nas` e `ontap-nas-flexgroup` drivers.

- Trident não oferece suporte à importação de volumes usando o `ontap-nas-economy` driver.
- Os `ontap-nas` e `ontap-nas-flexgroup` drivers não permitem nomes de volume duplicados.

Cada volume criado com o `ontap-nas` driver é um volume FlexVol no cluster ONTAP. Importar volumes FlexVol com o `ontap-nas` driver funciona da mesma forma. Um volume FlexVol que já existe em um cluster ONTAP pode ser importado como um `ontap-nas` PVC. Da mesma forma, volumes FlexGroup podem ser importados como `ontap-nas-flexgroup` PVCs.

Exemplos de ONTAP NAS usando tridentctl

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando `tridentctl`.

Volume gerenciado

O exemplo a seguir importa um volume chamado `managed_volume` em um backend chamado `ontap_nas`:

```
tridentctl import volume ontap_nas managed_volume -f <path-to-pvc-file>

+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-bf5ad463-afbb-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | true    |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

Volume não gerenciado

Ao usar o `--no-manage` argumento, Trident não renomeia o volume.

O exemplo a seguir importa `unmanaged_volume`no `ontap_nas`backend:

```
tridentctl import volume nas_blog unmanaged_volume -f <path-to-pvc-
file> --no-manage

+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-df07d542-afbc-11e9-8d9f-5254004dfdb7 | 1.0 GiB | standard      |
file     | c5a6f6a4-b052-423b-80d4-8fb491a14a22 | online | false   |
+----------------------------------------+---------+---------------
+----------+----------------------------------------+--------+---------+
```

Exemplos do ONTAP NAS usando anotações PVC

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados usando anotações PVC.

Volume gerenciado

O exemplo a seguir importa um volume de 1GiB `ontap-nas` nomeado `ontap_volume1` do backend `81abcb27-ea63-49bb-b606-0a5315ac5f21` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <managed-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap_volume1"
    trident.netapp.io/importBackendUUID: "81abcb27-ea63-49bb-b606-
0a5315ac5f21"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

Volume não gerenciado

O exemplo a seguir importa 1Gi `ontap-nas` volume nomeado `ontap-volume2` do backend `34abcb27-ea63-49bb-b606-0a5315ac5f34` com o modo de acesso RWO definido usando anotações PVC:

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: <umanaged-imported-volume>
  namespace: <namespace>
  annotations:
    trident.netapp.io/importOriginalName: "ontap-volume2"
    trident.netapp.io/importBackendUUID: "34abcb27-ea63-49bb-b606-
0a5315ac5f34"
    trident.netapp.io/notManaged: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: <storage-class-name>
```

ONTAP SAN

Trident suporta a importação de volumes usando os `ontap-san` (iSCSI, NVMe/TCP e FC) e `ontap-san-economy`drivers.

Trident pode importar volumes ONTAP SAN FlexVol que contêm um único LUN. Isso é consistente com o `ontap-san` driver, que cria um volume FlexVol para cada PVC e um LUN dentro do volume FlexVol. Trident importa o volume FlexVol e o associa à definição do PVC. Trident pode importar `ontap-san-economy` volumes que contêm múltiplos LUNs.

Os exemplos a seguir mostram como importar volumes gerenciados e não gerenciados:

Volume gerenciado

Para volumes gerenciados, Trident renomeia o FlexVol volume para o `pvc-<uuid>` formato e o LUN dentro do FlexVol volume para `lun0`.

O exemplo a seguir importa o `ontap-san-managed` FlexVol volume que está presente no `ontap_san_default` backend:

```
tridentctl import volume ontapsan_san_default ontap-san-managed -f pvc-
basic-import.yaml -n trident -d

+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-d6ee4f54-4e40-4454-92fd-d00fc228d74a | 20 MiB | basic         |
block    | cd394786-ddd5-4470-adc3-10c5ce4ca757 | online | true    |
+-----------------------------------------+--------+--------------
+----------+-----------------------------------------+--------+---------+
```

Volume não gerenciado

O exemplo a seguir importa `unmanaged_example_volume`no `ontap_san`backend:

```
tridentctl import volume -n trident san_blog unmanaged_example_volume
-f pvc-import.yaml --no-manage
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
|                   NAME                  |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
| pvc-1fc999c9-ce8c-459c-82e4-ed4380a4b228 | 1.0 GiB | san-blog      |
block    | e3275890-7d80-4af6-90cc-c7a0759f555a | online | false   |
+-----------------------------------------+---------+--------------
+----------+-----------------------------------------+--------+---------+
```

Se você tiver LUNs mapeadas para igroups que compartilham um IQN com o IQN de um nó do Kubernetes, como mostrado no exemplo a seguir, você receberá o erro: `LUN already mapped to initiator(s) in this group`. Você precisará remover o iniciador ou desmapear a LUN para importar o volume.

```
Vserver   Igroup       Protocol OS Type  Initiators
--------- ------------ -------- -------- ------------------------------------
svm0      k8s-nodename.example.com-fe5d36f2-cded-4f38-9eb0-c7719fc2f9f3
                       iscsi    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
svm0      unmanaged-example-igroup
                       mixed    linux    iqn.1994-05.com.redhat:4c2e1cf35e0
```

Elemento

Trident oferece suporte ao software NetApp Element e à importação de volumes NetApp HCI usando o driver `solidfire-san`.

> O driver Element suporta nomes de volume duplicados. No entanto, Trident retorna um erro se houver nomes de volume duplicados. Como solução alternativa, clone o volume, forneça um nome de volume exclusivo e importe o volume clonado.

O exemplo a seguir importa um `element-managed` volume no backend `element_default`.

```
tridentctl import volume element_default element-managed -f pvc-basic-
import.yaml -n trident -d

+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE  | STORAGE CLASS |
PROTOCOL |             BACKEND UUID              | STATE  | MANAGED |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
| pvc-970ce1ca-2096-4ecd-8545-ac7edc24a8fe | 10 GiB | basic-element |
block    | d3ba047a-ea0b-43f9-9c42-e38e58301c49 | online | true    |
+----------------------------------------+--------+---------------
+----------+----------------------------------------+--------+---------+
```

Azure NetApp Files

Trident oferece suporte à importação de volumes usando o `azure-netapp-files` driver.

> Para importar um volume do Azure NetApp Files, identifique o volume pelo seu caminho de volume. O caminho do volume é a parte do caminho de exportação do volume após o `:/`. Por exemplo, se o caminho de montagem for `10.0.0.2:/importvol1`, o caminho do volume é `importvol1`.

O exemplo a seguir importa um `azure-netapp-files` volume no backend `azurenetappfiles_40517` com o caminho do volume `importvol1`.

```
tridentctl import volume azurenetappfiles_40517 importvol1 -f <path-to-
pvc-file> -n trident

+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS |
PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
| pvc-0ee95d60-fd5c-448d-b505-b72901b3a4ab | 100 GiB | anf-storage   |
file     | 1c01274f-d94b-44a3-98a3-04c953c9a51e | online | true    |
+----------------------------------------+---------+--------------
+----------+----------------------------------------+--------+---------+
```

Google Cloud NetApp Volumes

Trident oferece suporte à importação de volumes usando o `google-cloud-netapp-volumes` driver.

O exemplo a seguir importa um volume no backend `backend-tbc-gcnv1` com o volume `testvoleasiaeast1`.

```
tridentctl import volume backend-tbc-gcnv1 "testvoleasiaeast1" -f < path-
to-pvc> -n trident

+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                 |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+----------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

O exemplo a seguir importa um `google-cloud-netapp-volumes` volume quando dois volumes estão presentes na mesma região:

```
tridentctl import volume backend-tbc-gcnv1
"projects/123456789100/locations/asia-east1-a/volumes/testvoleasiaeast1"
-f <path-to-pvc> -n trident

+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
|                   NAME                   |  SIZE   | STORAGE CLASS
| PROTOCOL |             BACKEND UUID             | STATE  | MANAGED |
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
| pvc-a69cda19-218c-4ca9-a941-aea05dd13dc0 |  10 GiB | gcnv-nfs-sc-
identity | file     | 8c18cdf1-0770-4bc0-bcc5-c6295fe6d837 | online | true
|
+------------------------------------------+---------
+----------------------+----------+--------------------------------------
+--------+---------+
```

Personalize os nomes e rótulos dos volumes

Com Trident, você pode atribuir nomes e rótulos significativos aos volumes que cria. Isso ajuda você a identificar e mapear facilmente os volumes aos seus respectivos recursos do Kubernetes (PVCs). Você também pode definir modelos no nível do backend para criar nomes de volumes personalizados e rótulos personalizados; quaisquer volumes que você criar, importar ou clonar seguirão os modelos.

Antes de começar

Suporte para nomes e rótulos de volume personalizáveis:

- Operações de criação, importação e clonagem de volume.
- No caso do `ontap-nas-economy`driver, apenas o nome do volume Qtree está em conformidade com o modelo de nome.
- No caso do `ontap-san-economy`driver, apenas o nome do LUN está em conformidade com o modelo de nome.

Limitações

- Os nomes de volumes personalizados são compatíveis apenas com drivers ONTAP locais.
- Rótulos personalizados são suportados apenas para os `ontap-san`, `ontap-nas` e `ontap-nas-flexgroup` drivers.
- Nomes de volume personalizados não se aplicam a volumes existentes.

Principais comportamentos dos nomes de volume personalizáveis

- Se ocorrer uma falha devido à sintaxe inválida em um modelo de nome, a criação do backend falhará. No entanto, se a aplicação do modelo falhar, o volume será nomeado de acordo com a convenção de nomenclatura existente.
- O prefixo de armazenamento não se aplica quando um volume é nomeado usando um modelo de nome da configuração do backend. Qualquer valor de prefixo desejado pode ser adicionado diretamente ao modelo.

Exemplos de configuração de backend com template de nome e rótulos

É possível definir modelos de nomes personalizados no nível raiz e/ou no nível do pool.

Exemplo de nível raiz

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "defaults": {
    "nameTemplate":
"{{.volume.Name}}_{{.labels.cluster}}_{{.volume.Namespace}}_{{.volume.Requ
estName}}"
  },
  "labels": {
    "cluster": "ClusterA",
    "PVC": "{{.volume.Namespace}}_{{.volume.RequestName}}"
  }
}
```

Exemplo de nível de pool

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "backendName": "ontap-nfs-backend",
  "managementLIF": "<ip address>",
  "svm": "svm0",
  "username": "<admin>",
  "password": "<password>",
  "useREST": true,
  "storage": [
    {
      "labels": {
        "labelname": "label1",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool01_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    },
    {
      "labels": {
        "cluster": "label2",
        "name": "{{ .volume.Name }}"
      },
      "defaults": {
        "nameTemplate": "pool02_{{ .volume.Name }}_{{ .labels.cluster
}}_{{ .volume.Namespace }}_{{ .volume.RequestName }}"
      }
    }
  ]
}
```

Exemplos de modelos de nome

Exemplo 1:

```
"nameTemplate": "{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
.config.BackendName }}"
```

Exemplo 2:

```
"nameTemplate": "pool_{{ .config.StoragePrefix }}_{{ .volume.Name }}_{{
slice .volume.RequestName 1 5 }}""
```

Pontos a considerar

1. No caso de importações de volume, os rótulos são atualizados somente se o volume existente possuir rótulos em um formato específico. Por exemplo: `{"provisioning":{"Cluster":"ClusterA", "PVC": "pvcname"}}`.
2. No caso de importações de volumes gerenciados, o nome do volume segue o modelo de nome definido no nível raiz na definição do backend.
3. Trident não suporta o uso de um operador de slice com o prefixo de armazenamento.
4. Se os modelos não resultarem em nomes de volume exclusivos, Trident adicionará alguns caracteres aleatórios para criar nomes de volume exclusivos.
5. Se o nome personalizado para um volume econômico do NAS exceder 64 caracteres, Trident nomeará os volumes de acordo com a convenção de nomenclatura existente. Para todos os outros drivers ONTAP, se o nome do volume exceder o limite de nomes, o processo de criação do volume falhará.

Compartilhe um volume NFS entre namespaces

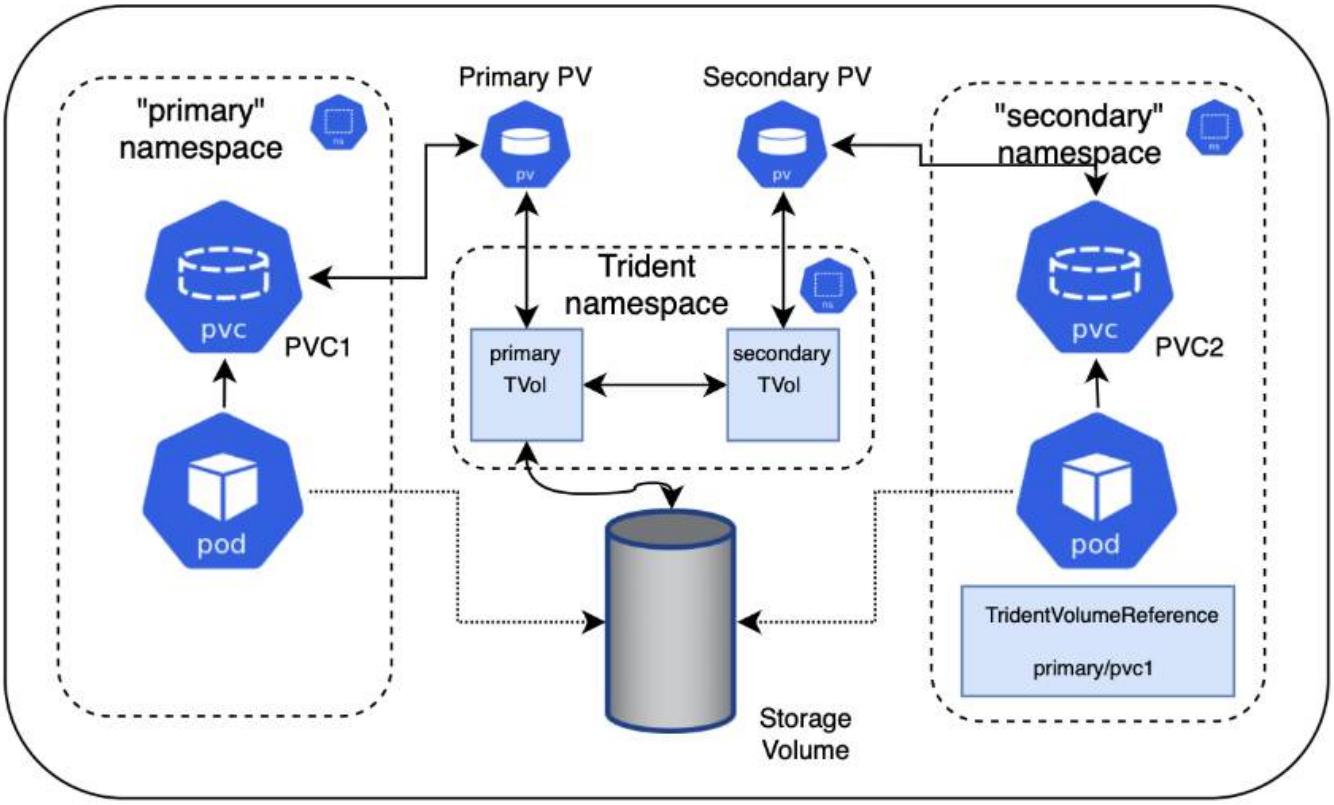
Usando Trident, você pode criar um volume em um namespace primário e compartilhá-lo em um ou mais namespaces secundários.

Características

O CR TridentVolumeReference permite que você compartilhe com segurança volumes NFS ReadWriteMany (RWX) entre um ou mais namespaces do Kubernetes. Essa solução nativa do Kubernetes oferece os seguintes benefícios:

- Múltiplos níveis de controle de acesso para garantir a segurança
- Compatível com todos os drivers de volume Trident NFS
- Sem dependência do tridentctl ou de qualquer outro recurso não nativo do Kubernetes

Este diagrama ilustra o compartilhamento de volumes NFS entre dois namespaces do Kubernetes.



Início rápido

Você pode configurar o compartilhamento de volumes NFS em apenas alguns passos.

1 **Configure o PVC de origem para compartilhar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 **Conceda permissão para criar um CR no namespace de destino**

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR.

3 **Criar TridentVolumeReference no namespace de destino**

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência ao PVC de origem.

4 **Crie o PVC subordinado no namespace de destino**

O proprietário do namespace de destino cria o PVC subordinado para usar a fonte de dados do PVC de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, o compartilhamento entre namespaces exige colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1` no namespace de origem que concede permissão para compartilhar com o namespace de destino (`namespace2` usando a anotação `shareToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/shareToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident cria o PV e seu volume de armazenamento NFS de backend.

   - Você pode compartilhar o PVC com vários namespaces usando uma lista separada por vírgulas. Por exemplo, `trident.netapp.io/shareToNamespace: namespace2,namespace3,namespace4`.
   - Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/shareToNamespace: *`
   - Você pode atualizar o PVC para incluir a `shareToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja implementado para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino.
3. **Proprietário do namespace de destino:** Crie um CR TridentVolumeReference no namespace de destino que faça referência ao namespace de origem `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando a anotação `shareFromPVC` para designar o PVC de origem.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/shareFromPVC: namespace1/pvc1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

O tamanho do PVC de destino deve ser menor ou igual ao do PVC de origem.

Resultados

Trident lê a `shareFromPVC` anotação no PVC de destino e cria o PV de destino como um volume subordinado, sem recurso de armazenamento próprio, que aponta para o PV de origem e compartilha o recurso de armazenamento do PV de origem. O PVC de destino e o PV de destino aparecem vinculados normalmente.

Excluir um volume compartilhado

Você pode excluir um volume compartilhado entre vários namespaces. Trident removerá o acesso ao volume no namespace de origem e manterá o acesso para outros namespaces que compartilham o volume. Quando todos os namespaces que fazem referência ao volume são removidos, Trident exclui o volume.

Use `tridentctl get` para consultar volumes subordinados

Usando o[`tridentctl` utilitário, você pode executar o `get` comando para obter volumes subordinados. Para mais informações, consulte `tridentctl` comandos e opções.

```
Usage:
  tridentctl get [option]
```

Bandeiras:

- `-h, --help`: Ajuda para volumes.
- `--parentOfSubordinate string`: limitar a consulta ao volume de origem subordinado.
- `--subordinateOf string`: limitar a consulta aos subordinados do volume.

Limitações

- Trident não pode impedir que os namespaces de destino gravem no volume compartilhado. Você deve usar bloqueio de arquivos ou outros processos para evitar a sobrescrita de dados do volume compartilhado.
- Não é possível revogar o acesso ao PVC de origem removendo as `shareToNamespace` ou `shareFromNamespace` anotações ou excluindo o `TridentVolumeReference` CR. Para revogar o acesso, é necessário excluir o PVC subordinado.
- Snapshots, clones e espelhamento não são possíveis em volumes subordinados.

Para obter mais informações

Para saber mais sobre acesso a volumes entre namespaces:

- Visite "Compartilhamento de volumes entre namespaces: diga olá ao acesso a volumes entre namespaces".
- Veja a demonstração em "NetAppTV".

Clonar volumes entre namespaces

Usando Trident, você pode criar novos volumes usando volumes existentes ou volumesnapshots de um namespace diferente dentro do mesmo cluster Kubernetes.

Pré-requisitos

Antes de clonar volumes, certifique-se de que os backends de origem e destino sejam do mesmo tipo e tenham a mesma classe de armazenamento.

> A clonagem entre namespaces é suportada apenas para os `ontap-san` e `ontap-nas` drivers de armazenamento. Clones somente leitura não são suportados.

Início rápido

Você pode configurar a clonagem de volumes em apenas alguns passos.

1 **Configure o PVC de origem para clonar o volume**

O proprietário do namespace de origem concede permissão para acessar os dados no PVC de origem.

2 Conceda permissão para criar um CR no namespace de destino

O administrador do cluster concede permissão ao proprietário do namespace de destino para criar o TridentVolumeReference CR.

3 Criar TridentVolumeReference no namespace de destino

O proprietário do namespace de destino cria o TridentVolumeReference CR para fazer referência ao PVC de origem.

4 Crie o PVC clonado no namespace de destino

O proprietário do namespace de destino cria um PVC para clonar o PVC do namespace de origem.

Configure os namespaces de origem e destino

Para garantir a segurança, a clonagem de volumes entre namespaces requer colaboração e ação do proprietário do namespace de origem, do administrador do cluster e do proprietário do namespace de destino. A função do usuário é designada em cada etapa.

Passos

1. **Proprietário do namespace de origem:** Crie o PVC (`pvc1`) no namespace de origem (`namespace1`) que concede permissão para compartilhar com o namespace de destino (`namespace2`) usando a anotação `cloneToNamespace`.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1
  namespace: namespace1
  annotations:
    trident.netapp.io/cloneToNamespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

   Trident cria o PV e seu volume de armazenamento backend.

- Você pode compartilhar o PVC com vários namespaces usando uma lista separada por vírgulas. Por exemplo, `trident.netapp.io/cloneToNamespace: namespace2,namespace3,namespace4`.
- Você pode compartilhar com todos os namespaces usando `*`. Por exemplo, `trident.netapp.io/cloneToNamespace: *`
- Você pode atualizar o PVC para incluir a `cloneToNamespace` anotação a qualquer momento.

2. **Administrador do cluster:** Certifique-se de que o RBAC adequado esteja configurado para conceder permissão ao proprietário do namespace de destino para criar o CR TridentVolumeReference no namespace de destino (`namespace2`).
3. **Proprietário do namespace de destino:** Crie um CR TridentVolumeReference no namespace de destino que faça referência ao namespace de origem `pvc1`.

```
apiVersion: trident.netapp.io/v1
kind: TridentVolumeReference
metadata:
  name: my-first-tvr
  namespace: namespace2
spec:
  pvcName: pvc1
  pvcNamespace: namespace1
```

4. **Proprietário do namespace de destino:** Crie um PVC (`pvc2`) no namespace de destino (`namespace2`) usando as `cloneFromPVC` ou `cloneFromSnapshot`, e `cloneFromNamespace` anotações para designar o PVC de origem.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  annotations:
    trident.netapp.io/cloneFromPVC: pvc1
    trident.netapp.io/cloneFromNamespace: namespace1
  name: pvc2
  namespace: namespace2
spec:
  accessModes:
    - ReadWriteMany
  storageClassName: trident-csi
  resources:
    requests:
      storage: 100Gi
```

Limitações

- Para PVCs provisionados usando drivers ontap-nas-economy, clones somente leitura não são suportados.

Replicar volumes usando SnapMirror

Trident oferece suporte a relações de espelhamento entre um volume de origem em um cluster e o volume de destino no cluster emparelhado para replicação de dados em casos de recuperação de desastres. Você pode usar uma Definição de Recurso Personalizado (CRD) com namespace, chamada Trident Mirror Relationship (TMR), para realizar as seguintes operações:

- Criar relações de espelhamento entre volumes (PVCs)
- Remova relações de espelhamento entre volumes
- Interrompa os relacionamentos espelhados
- Promover o volume secundário em situações de desastre (falhas)
- Realizar a transição sem perda de dados de aplicações de um cluster para outro (durante failovers ou migrações planejadas)

Pré-requisitos de replicação

Certifique-se de que os seguintes pré-requisitos sejam atendidos antes de começar:

Clusters do ONTAP

- **Trident**: A versão 22.10 ou posterior do Trident deve existir em ambos os clusters Kubernetes de origem e destino que utilizam ONTAP como backend.
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote Data Protection devem estar habilitadas tanto nos clusters ONTAP de origem quanto de destino. Consulte "SnapMirror visão geral do licenciamento no ONTAP" para mais informações.

  A partir do ONTAP 9.10.1, todas as licenças são fornecidas como um arquivo de licença NetApp (NLF), que é um único arquivo que habilita vários recursos. Consulte "Licenças incluídas com o ONTAP One" para mais informações.

  > Apenas a proteção assíncrona SnapMirror é suportada.

Peering

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar emparelhados. Consulte "Visão geral do peering de cluster e SVM" para mais informações.

  > Certifique-se de que os nomes SVM usados na relação de replicação entre dois clusters ONTAP sejam únicos.

- **Trident e SVM**: Os SVMs remotos emparelhados devem estar disponíveis para Trident no cluster de destino.

Drivers com suporte

NetApp Trident suporta replicação de volumes com a tecnologia NetApp SnapMirror usando classes de armazenamento com suporte dos seguintes drivers: **`ontap-nas`: NFS** `ontap-san`: iSCSI **`ontap-san`: FC**

`ontap-san`: NVMe/TCP (requer versão mínima do ONTAP 9.15.1)

> A replicação de volume usando SnapMirror não é suportada para sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte "Saiba mais sobre os sistemas de armazenamento ASA r2".

Crie um PVC espelhado

Siga estes passos e utilize os exemplos de CRD para criar relação de espelhamento entre os volumes primário e secundário.

Passos

1. Execute as seguintes etapas no cluster Kubernetes primário:
   a. Crie um objeto StorageClass com o parâmetro `trident.netapp.io/replication: true`.

      Exemplo

      ```
      apiVersion: storage.k8s.io/v1
      kind: StorageClass
      metadata:
        name: csi-nas
      provisioner: csi.trident.netapp.io
      parameters:
        backendType: "ontap-nas"
        fsType: "nfs"
        trident.netapp.io/replication: "true"
      ```

   b. Crie um PVC com StorageClass previamente criado.

      Exemplo

      ```
      kind: PersistentVolumeClaim
      apiVersion: v1
      metadata:
        name: csi-nas
      spec:
        accessModes:
        - ReadWriteMany
        resources:
          requests:
            storage: 1Gi
        storageClassName: csi-nas
      ```

   c. Crie um MirrorRelationship CR com informações locais.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
```

Trident obtém as informações internas do volume e o estado atual de proteção de dados (DP) do volume, em seguida, preenche o campo de status do MirrorRelationship.

d. Obtenha o TridentMirrorRelationship CR para obter o nome interno e o SVM do PVC.

```
kubectl get tmr csi-nas
```

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
  generation: 1
spec:
  state: promoted
  volumeMappings:
  - localPVCName: csi-nas
status:
  conditions:
  - state: promoted
    localVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
    localPVCName: csi-nas
    observedGeneration: 1
```

2. Execute as seguintes etapas no cluster Kubernetes secundário:

   a. Crie um StorageClass com o parâmetro trident.netapp.io/replication: true.

Exemplo

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-nas
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/replication: true
```

b. Crie um CR MirrorRelationship com informações de destino e origem.

Exemplo

```
kind: TridentMirrorRelationship
apiVersion: trident.netapp.io/v1
metadata:
  name: csi-nas
spec:
  state: established
  volumeMappings:
  - localPVCName: csi-nas
    remoteVolumeHandle:
"datavserver:trident_pvc_3bedd23c_46a8_4384_b12b_3c38b313c1e1"
```

Trident criará um SnapMirror relacionamento com o nome da política de relacionamento configurada (ou padrão para ONTAP) e o inicializará.

c. Crie um PVC com o StorageClass previamente criado para atuar como secundário (SnapMirror destino).

Exemplo

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: csi-nas
  annotations:
    trident.netapp.io/mirrorRelationship: csi-nas
spec:
  accessModes:
  - ReadWriteMany
resources:
  requests:
    storage: 1Gi
storageClassName: csi-nas
```

Trident verificará o CRD TridentMirrorRelationship e não conseguirá criar o volume se o relacionamento não existir. Se o relacionamento existir, Trident garantirá que o novo volume FlexVol seja colocado em uma SVM que esteja emparelhada com a SVM remota definida no MirrorRelationship.

Estados de replicação de volume

Um Trident Mirror Relationship (TMR) é um CRD que representa uma das extremidades de uma relação de replicação entre PVCs. O TMR de destino possui um estado, que informa ao Trident qual é o estado desejado. O TMR de destino possui os seguintes estados:

- **Estabelecido**: o PVC local é o volume de destino de uma relação de espelhamento, e esta é uma nova relação.
- **Promovido**: o PVC local é ReadWrite e montável, sem relação de espelhamento em vigor no momento.
- **Reestabelecido**: o PVC local é o volume de destino de uma relação de espelhamento e também estava anteriormente nessa relação de espelhamento.
  - O estado restabelecido deve ser usado se o volume de destino já teve alguma relação com o volume de origem, pois ele sobrescreve o conteúdo do volume de destino.
  - O estado restabelecido falhará se o volume não estava previamente em uma relação com a origem.

Promover o PVC secundário durante uma falha não planejada

Execute a seguinte etapa no cluster Kubernetes secundário:

- Atualize o campo *spec.state* de TridentMirrorRelationship para `promoted`.

Promover o PVC secundário durante uma falha planejada

Durante um failover (migração) planejado, execute as seguintes etapas para promover o PVC secundário:

Passos

1. No cluster Kubernetes primário, crie um snapshot do PVC e aguarde até que o snapshot seja criado.
2. No cluster Kubernetes primário, crie o SnapshotInfo CR para obter detalhes internos.

   Exemplo

   ```
   kind: SnapshotInfo
   apiVersion: trident.netapp.io/v1
   metadata:
     name: csi-nas
   spec:
     snapshot-name: csi-nas-snapshot
   ```

3. No cluster Kubernetes secundário, atualize o campo *spec.state* do CR *TridentMirrorRelationship* para *promoted* e *spec.promotedSnapshotHandle* para ser o internalName do snapshot.
4. No cluster Kubernetes secundário, confirme o status (campo status.state) de TridentMirrorRelationship para promovido.

Restaurar uma relação de espelhamento após uma falha

Antes de restaurar a relação de espelhamento, escolha o lado que deseja definir como o novo primário.

Passos

1. No cluster Kubernetes secundário, verifique se os valores do campo *spec.remoteVolumeHandle* em TridentMirrorRelationship foram atualizados.
2. No cluster Kubernetes secundário, atualize o campo *spec.mirror* de TridentMirrorRelationship para `reestablished`.

Operações adicionais

Trident suporta as seguintes operações nos volumes primário e secundário:

Replicar o PVC primário em um novo PVC secundário

Certifique-se de que você já possui um PVC primário e um PVC secundário.

Passos

1. Exclua os CRDs PersistentVolumeClaim e TridentMirrorRelationship do cluster secundário (destino) estabelecido.
2. Exclua o CRD TridentMirrorRelationship do cluster primário (de origem).
3. Crie um novo TridentMirrorRelationship CRD no cluster primário (origem) para o novo PVC secundário (destino) que você deseja estabelecer.

Redimensione um PVC espelhado, primário ou secundário

O PVC pode ser redimensionado normalmente, o ONTAP expandirá automaticamente qualquer FlexVol volume de destino se a quantidade de dados exceder o tamanho atual.

Remover replicação de um PVC

Para remover a replicação, execute uma das seguintes operações no volume secundário atual:

- Exclua o MirrorRelationship no PVC secundário. Isso interrompe a relação de replicação.
- Ou, atualize o campo spec.state para *promoted*.

Excluir um PVC (que foi previamente espelhado)

Trident verifica se há PVCs replicados e libera a relação de replicação antes de tentar excluir o volume.

Excluir um TMR

A exclusão de um TMR em um dos lados de um relacionamento espelhado faz com que o TMR restante passe para o estado *promovido* antes que o Trident conclua a exclusão. Se o TMR selecionado para exclusão já estiver no estado *promovido*, não haverá um relacionamento espelhado existente e o TMR será removido e o Trident promoverá o PVC local para *ReadWrite*. Essa exclusão libera os metadados SnapMirror do volume local no ONTAP. Se esse volume for usado em um relacionamento espelhado no futuro, será necessário usar um novo TMR com um estado de replicação de volume *estabelecido* ao criar o novo relacionamento espelhado.

Atualize os relacionamentos de espelhamento quando ONTAP estiver online

As relações de espelhamento podem ser atualizadas a qualquer momento após serem estabelecidas. Você pode usar o `state: promoted` ou `state: reestablished` campos para atualizar as relações. Ao promover um volume de destino para um volume ReadWrite regular, você pode usar *promotedSnapshotHandle* para especificar um snapshot específico para restaurar o volume atual.

Atualizar relações de espelhamento quando ONTAP estiver offline

Você pode usar um CRD para realizar uma atualização SnapMirror sem que o Trident tenha conectividade direta com o cluster ONTAP. Consulte o seguinte exemplo de formato do TridentActionMirrorUpdate:

Exemplo

```
apiVersion: trident.netapp.io/v1
kind: TridentActionMirrorUpdate
metadata:
  name: update-mirror-b
spec:
  snapshotHandle: "pvc-1234/snapshot-1234"
  tridentMirrorRelationshipName: mirror-b
```

`status.state` reflete o estado do TridentActionMirrorUpdate CRD. Pode assumir um valor de *Succeeded*, *In Progress* ou *Failed*.

Usar a topologia CSI

Trident pode criar e anexar volumes seletivamente aos nós presentes em um cluster Kubernetes fazendo uso do "Recurso de topologia CSI".

Visão geral

Utilizando o recurso de CSI Topology, o acesso a volumes pode ser limitado a um subconjunto de nós, com base em regiões e zonas de disponibilidade. Atualmente, os provedores de nuvem permitem que os administradores do Kubernetes criem nós baseados em zonas. Os nós podem estar localizados em diferentes zonas de disponibilidade dentro de uma região ou em várias regiões. Para facilitar o provisionamento de volumes para cargas de trabalho em uma arquitetura multizona, Trident utiliza CSI Topology.

Saiba mais sobre o recurso de topologia CSI "aqui".

Kubernetes oferece dois modos exclusivos de vinculação de volumes:

- Com `VolumeBindingMode` definido como `Immediate`, Trident cria o volume sem qualquer reconhecimento de topologia. A vinculação de volume e o provisionamento dinâmico são tratados quando o PVC é criado. Este é o padrão `VolumeBindingMode` e é adequado para clusters que não impõem restrições de topologia. Volumes Persistentes são criados sem depender dos requisitos de agendamento do pod solicitante.
- Com `VolumeBindingMode` definido como `WaitForFirstConsumer`, a criação e a vinculação de um Volume Persistente para um PVC são adiadas até que um pod que utiliza o PVC seja agendado e criado. Dessa forma, os volumes são criados para atender às restrições de agendamento impostas pelos requisitos de topologia.

O `WaitForFirstConsumer`modo de vinculação não requer rótulos de topologia. Isso pode ser usado independentemente do recurso de topologia CSI.

O que você vai precisar

Para utilizar a topologia CSI, você precisa do seguinte:

- Um cluster Kubernetes executando um "versão do Kubernetes suportada"

```
kubectl version
Client Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:50:19Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
Server Version: version.Info{Major:"1", Minor:"19",
GitVersion:"v1.19.3",
GitCommit:"1e11e4a2108024935ecfcb2912226cedeafd99df",
GitTreeState:"clean", BuildDate:"2020-10-14T12:41:49Z",
GoVersion:"go1.15.2", Compiler:"gc", Platform:"linux/amd64"}
```

- Os nós do cluster devem ter rótulos que introduzam a consciência de topologia (`topology.kubernetes.io/region` e `topology.kubernetes.io/zone`). Esses rótulos **devem estar presentes nos nós do cluster** antes que o Trident seja instalado para que o Trident reconheça a topologia.

```
kubectl get nodes -o=jsonpath='{range .items[*]}[{.metadata.name},
{.metadata.labels}]{"\n"}{end}' | grep --color "topology.kubernetes.io"
[node1,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node1","kubernetes.io/
os":"linux","node-
role.kubernetes.io/master":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-a"}]
[node2,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node2","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-b"}]
[node3,
{"beta.kubernetes.io/arch":"amd64","beta.kubernetes.io/os":"linux","kube
rnetes.io/arch":"amd64","kubernetes.io/hostname":"node3","kubernetes.io/
os":"linux","node-
role.kubernetes.io/worker":"","topology.kubernetes.io/region":"us-
east1","topology.kubernetes.io/zone":"us-east1-c"}]
```

Etapa 1: crie um backend com reconhecimento de topologia

Os backends de storage Trident podem ser projetados para provisionar volumes seletivamente com base em zonas de disponibilidade. Cada backend pode conter um bloco `supportedTopologies` opcional que representa uma lista de zonas e regiões suportadas. Para StorageClasses que fazem uso de tal backend, um volume só será criado se solicitado por uma aplicação agendada em uma região/zona suportada.

Aqui está um exemplo de definição de backend:

YAML

```
---
version: 1
storageDriverName: ontap-san
backendName: san-backend-us-east1
managementLIF: 192.168.27.5
svm: iscsi_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-a
  - topology.kubernetes.io/region: us-east1
    topology.kubernetes.io/zone: us-east1-b
```

JSON

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "backendName": "san-backend-us-east1",
  "managementLIF": "192.168.27.5",
  "svm": "iscsi_svm",
  "username": "admin",
  "password": "password",
  "supportedTopologies": [
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-a"
    },
    {
      "topology.kubernetes.io/region": "us-east1",
      "topology.kubernetes.io/zone": "us-east1-b"
    }
  ]
}
```

`supportedTopologies` é usado para fornecer uma lista de regiões e zonas por backend. Essas regiões e zonas representam a lista de valores permitidos que podem ser fornecidos em um StorageClass. Para StorageClasses que contêm um subconjunto das regiões e zonas fornecidas em um backend, Trident cria um volume no backend.

Você pode definir `supportedTopologies` por pool de storage também. Veja o exemplo a seguir:

```
---
version: 1
storageDriverName: ontap-nas
backendName: nas-backend-us-central1
managementLIF: 172.16.238.5
svm: nfs_svm
username: admin
password: password
supportedTopologies:
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-a
  - topology.kubernetes.io/region: us-central1
    topology.kubernetes.io/zone: us-central1-b
storage:
  - labels:
      workload: production
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-a
  - labels:
      workload: dev
    supportedTopologies:
      - topology.kubernetes.io/region: us-central1
        topology.kubernetes.io/zone: us-central1-b
```

Neste exemplo, os `region` e `zone` rótulos representam a localização do pool de storage. `topology.kubernetes.io/region` e `topology.kubernetes.io/zone` indicam de onde os pools de storage podem ser consumidos.

Etapa 2: Defina StorageClasses que são cientes da topologia

Com base nos rótulos de topologia fornecidos aos nós do cluster, StorageClasses podem ser definidos para conter informações de topologia. Isso determinará os pools de storage que servem como candidatos para solicitações de PVC feitas, e o subconjunto de nós que podem fazer uso dos volumes provisionados pelo Trident.

Veja o seguinte exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: netapp-san-us-east1
provisioner: csi.trident.netapp.io
volumeBindingMode: WaitForFirstConsumer
allowedTopologies:
  - matchLabelExpressions:
    - key: topology.kubernetes.io/zone
      values:
        - us-east1-a
        - us-east1-b
    - key: topology.kubernetes.io/region
      values:
        - us-east1
parameters:
  fsType: ext4
```

Na definição de StorageClass fornecida acima, `volumeBindingMode` está definido como `WaitForFirstConsumer`. Os PVCs solicitados com este StorageClass não serão processados até que sejam referenciados em um pod. E, `allowedTopologies` fornece as zonas e a região a serem usadas. O `netapp-san-us-east1` StorageClass cria PVCs no backend `san-backend-us-east1` definido acima.

Etapa 3: criar e usar um PVC

Com o StorageClass criado e mapeado para um backend, agora você pode criar PVCs.

Veja o exemplo `spec` abaixo:

```
---
kind: PersistentVolumeClaim
apiVersion: v1
metadata: null
name: pvc-san
spec: null
accessModes:
  - ReadWriteOnce
resources:
  requests:
    storage: 300Mi
storageClassName: netapp-san-us-east1
```

A criação de um PVC usando este manifesto resultaria no seguinte:

```
kubectl create -f pvc.yaml
persistentvolumeclaim/pvc-san created
kubectl get pvc
NAME      STATUS    VOLUME   CAPACITY   ACCESS MODES   STORAGECLASS
AGE
pvc-san   Pending                                      netapp-san-us-east1
2s
kubectl describe pvc
Name:          pvc-san
Namespace:     default
StorageClass:  netapp-san-us-east1
Status:        Pending
Volume:
Labels:        <none>
Annotations:   <none>
Finalizers:    [kubernetes.io/pvc-protection]
Capacity:
Access Modes:
VolumeMode:    Filesystem
Mounted By:    <none>
Events:
  Type    Reason                Age   From                         Message
  ----    ------                ----  ----                         -------
  Normal  WaitForFirstConsumer  6s    persistentvolume-controller  waiting
for first consumer to be created before binding
```

Para que Trident crie um volume e o vincule ao PVC, use o PVC em um pod. Veja o exemplo a seguir:

```
apiVersion: v1
kind: Pod
metadata:
  name: app-pod-1
spec:
  affinity:
    nodeAffinity:
      requiredDuringSchedulingIgnoredDuringExecution:
        nodeSelectorTerms:
        - matchExpressions:
          - key: topology.kubernetes.io/region
            operator: In
            values:
            - us-east1
      preferredDuringSchedulingIgnoredDuringExecution:
      - weight: 1
        preference:
          matchExpressions:
          - key: topology.kubernetes.io/zone
            operator: In
            values:
            - us-east1-a
            - us-east1-b
  securityContext:
    runAsUser: 1000
    runAsGroup: 3000
    fsGroup: 2000
  volumes:
  - name: vol1
    persistentVolumeClaim:
      claimName: pvc-san
  containers:
  - name: sec-ctx-demo
    image: busybox
    command: [ "sh", "-c", "sleep 1h" ]
    volumeMounts:
    - name: vol1
      mountPath: /data/demo
    securityContext:
      allowPrivilegeEscalation: false
```

Este podSpec instrui o Kubernetes a agendar o pod em nós que estão presentes na `us-east1` região e escolher entre qualquer nó que esteja presente nas `us-east1-a` ou `us-east1-b` zonas.

Veja a seguinte saída:

```
kubectl get pods -o wide
NAME        READY   STATUS    RESTARTS   AGE   IP               NODE
NOMINATED NODE   READINESS GATES
app-pod-1   1/1     Running   0          19s   192.168.25.131   node2
<none>           <none>
kubectl get pvc -o wide
NAME      STATUS   VOLUME                                     CAPACITY
ACCESS MODES   STORAGECLASS          AGE   VOLUMEMODE
pvc-san   Bound    pvc-ecb1e1a0-840c-463b-8b65-b3d033e2e62b   300Mi
RWO            netapp-san-us-east1   48s   Filesystem
```

Atualize os backends para incluir `supportedTopologies`

Os backends preexistentes podem ser atualizados para incluir uma lista de `supportedTopologies` usando `tridentctl backend update`. Isso não afetará os volumes que já foram provisionados e será usado apenas para PVCs subsequentes.

Encontre mais informações

- "Gerenciar recursos para contêineres"
- "nodeSelector"
- "Afinidade e antiafinidade"
- "Taints e Tolerations"

Trabalhar com Snapshots

Os snapshots de volumes persistentes (PVs) do Kubernetes permitem cópias de volumes em ponto no tempo. Você pode criar um snapshot de um volume criado usando Trident, importar um snapshot criado fora do Trident, criar um novo volume a partir de um snapshot existente e recuperar dados de volumes a partir de snapshots.

Visão geral

O snapshot de volume é suportado pelos `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san`, `azure-netapp-files` e `google-cloud-netapp-volumes` drivers.

Antes de começar

Você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, consulte Implante um controlador de Snapshot de volume.

> Não crie um controlador de snapshot se estiver criando snapshots de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

Criar um snapshot de volume

Passos

1. Crie um `VolumeSnapshotClass`. Para obter mais informações, consulte "VolumeSnapshotClass".
   - O `driver` aponta para o driver Trident CSI.
   - `deletionPolicy` pode ser `Delete` ou `Retain`. Quando definido como `Retain`, o snapshot físico subjacente no cluster de armazenamento é mantido mesmo quando o objeto `VolumeSnapshot` é excluído.

     Exemplo

     ```
     cat snap-sc.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshotClass
     metadata:
       name: csi-snapclass
     driver: csi.trident.netapp.io
     deletionPolicy: Delete
     ```

2. Crie um snapshot de um PVC existente.

   Exemplos

   - Este exemplo cria um snapshot de um PVC existente.

     ```
     cat snap.yaml
     ```

     ```
     apiVersion: snapshot.storage.k8s.io/v1
     kind: VolumeSnapshot
     metadata:
       name: pvc1-snap
     spec:
       volumeSnapshotClassName: csi-snapclass
       source:
         persistentVolumeClaimName: pvc1
     ```

   - Este exemplo cria um objeto de snapshot de volume para um PVC nomeado `pvc1` e o nome do snapshot é definido como `pvc1-snap`. Um VolumeSnapshot é análogo a um PVC e está associado a um `VolumeSnapshotContent` objeto que representa o snapshot real.

```
kubectl create -f snap.yaml
volumesnapshot.snapshot.storage.k8s.io/pvc1-snap created

kubectl get volumesnapshots
NAME                    AGE
pvc1-snap               50s
```

- Você pode identificar o `VolumeSnapshotContent` objeto para o `pvc1-snap` VolumeSnapshot descrevendo-o. O `Snapshot Content Name` identifica o objeto VolumeSnapshotContent que serve a este snapshot. O `Ready To Use` parâmetro indica que o snapshot pode ser usado para criar um novo PVC.

```
kubectl describe volumesnapshots pvc1-snap
Name:         pvc1-snap
Namespace:    default
...
Spec:
  Snapshot Class Name:    pvc1-snap
  Snapshot Content Name:  snapcontent-e8d8a0ca-9826-11e9-9807-
525400f3f660
  Source:
    API Group:
    Kind:       PersistentVolumeClaim
    Name:       pvc1
Status:
  Creation Time:  2019-06-26T15:27:29Z
  Ready To Use:   true
  Restore Size:   3Gi
...
```

Crie um PVC a partir de um instantâneo de volume

Você pode usar `dataSource` para criar um PVC usando um VolumeSnapshot chamado `<pvc-name>` como fonte dos dados. Depois que o PVC é criado, ele pode ser anexado a um pod e usado como qualquer outro PVC.

> O PVC será criado no mesmo backend que o volume de origem. Consulte "KB: A criação de um PVC a partir de um Trident PVC Snapshot não pode ser feita em um backend alternativo".

O exemplo a seguir cria o PVC usando `pvc1-snap` como fonte de dados.

```
cat pvc-from-snap.yaml
```

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: golden
  resources:
    requests:
      storage: 3Gi
  dataSource:
    name: pvc1-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Importar um Snapshot de volume

Trident oferece suporte à "Processo de snapshot pré-provisionado do Kubernetes" para permitir que o administrador do cluster crie um `VolumeSnapshotContent` objeto e importe snapshots criados fora do Trident.

Antes de começar

Trident deve ter criado ou importado o volume pai do Snapshot.

Passos

1. **Administrador do cluster:** Crie um `VolumeSnapshotContent` objeto que faça referência ao snapshot do backend. Isso inicia o fluxo de trabalho de snapshot no Trident.
   - Especifique o nome do snapshot do backend em `annotations` como `trident.netapp.io/internalSnapshotName: <"backend-snapshot-name">`.
   - Especifique `<name-of-parent-volume-in-trident>/<volume-snapshot-content-name>` em `snapshotHandle`. Esta é a única informação fornecida ao Trident pelo snapshotter externo na chamada `ListSnapshots`.

     > O `<volumeSnapshotContentName>` não pode sempre corresponder ao nome do snapshot do backend devido a restrições de nomenclatura do CR.

     Exemplo

     O exemplo a seguir cria um `VolumeSnapshotContent` objeto que faz referência ao snapshot do backend `snap-01`.

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotContent
metadata:
  name: import-snap-content
  annotations:
    trident.netapp.io/internalSnapshotName: "snap-01"  # This is the
name of the snapshot on the backend
spec:
  deletionPolicy: Retain
  driver: csi.trident.netapp.io
  source:
    snapshotHandle: pvc-f71223b5-23b9-4235-bbfe-e269ac7b84b0/import-
snap-content # <import PV name or source PV name>/<volume-snapshot-
content-name>
  volumeSnapshotRef:
    name: import-snap
    namespace: default
```

2. **Administrador do cluster:** Crie o `VolumeSnapshot` CR que referencia o `VolumeSnapshotContent` objeto. Isso solicita acesso para usar o `VolumeSnapshot` em um determinado namespace.

   Exemplo

   O exemplo a seguir cria um `VolumeSnapshot` CR chamado `import-snap` que faz referência ao `VolumeSnapshotContent` chamado `import-snap-content`.

   ```
   apiVersion: snapshot.storage.k8s.io/v1
   kind: VolumeSnapshot
   metadata:
     name: import-snap
   spec:
     # volumeSnapshotClassName: csi-snapclass (not required for pre-
   provisioned or imported snapshots)
     source:
       volumeSnapshotContentName: import-snap-content
   ```

3. **Processamento interno (nenhuma ação necessária):** O snapshotter externo reconhece o recém-criado `VolumeSnapshotContent` e executa a `ListSnapshots` chamada. Trident cria o `TridentSnapshot`.
   - O snapshotter externo define o `VolumeSnapshotContent` para `readyToUse` e o `VolumeSnapshot` para `true`.
   - Trident está de volta `readyToUse=true`.
4. **Qualquer usuário:** Crie um `PersistentVolumeClaim` para referenciar o novo `VolumeSnapshot`, onde o `spec.dataSource` (ou `spec.dataSourceRef`) nome é o nome `VolumeSnapshot`.

   Exemplo

O exemplo a seguir cria um PVC referenciando o `VolumeSnapshot` nomeado `import-snap`.

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: pvc-from-snap
spec:
  accessModes:
    - ReadWriteOnce
  storageClassName: simple-sc
  resources:
    requests:
      storage: 1Gi
  dataSource:
    name: import-snap
    kind: VolumeSnapshot
    apiGroup: snapshot.storage.k8s.io
```

Recupere dados de volume usando Snapshots

O diretório de snapshots está oculto por padrão para facilitar a máxima compatibilidade de volumes provisionados usando os `ontap-nas` e `ontap-nas-economy` drivers. Habilite o `.snapshot` diretório para recuperar dados diretamente dos snapshots.

Use o volume snapshot restore ONTAP CLI para restaurar um volume a um estado registrado em um snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Ao restaurar uma cópia de Snapshot, a configuração do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Trident suporta a restauração de snapshot nos `ontap-san`, `ontap-san-economy`, `ontap-nas`, `ontap-nas-flexgroup`, `azure-netapp-files`, `google-cloud-netapp-volumes` e `solidfire-san` drivers.

Antes de começar

Você deve ter um PVC vinculado e um snapshot de volume disponível.

- Verifique se o status do PVC está bound.

```
kubectl get pvc
```

- Verifique se o snapshot do volume está pronto para uso.

```
kubectl get vs
```

Passos

1. Crie o CR TASR. Este exemplo cria um CR para PVC `pvc1` e volume snapshot `pvc1-snapshot`.

   O TASR CR deve estar em um namespace onde o PVC e o VS existam.

```
cat tasr-pvc1-snapshot.yaml
```

```
apiVersion: trident.netapp.io/v1
kind: TridentActionSnapshotRestore
metadata:
  name: trident-snap
  namespace: trident
spec:
  pvcName: pvc1
  volumeSnapshotName: pvc1-snapshot
```

2. Aplique a CR para restaurar a partir do snapshot. Este exemplo restaura a partir do snapshot `pvc1`.

```
kubectl create -f tasr-pvc1-snapshot.yaml
```

```
tridentactionsnapshotrestore.trident.netapp.io/trident-snap created
```

Resultados

Trident restaura os dados a partir do snapshot. Você pode verificar o status da restauração do snapshot:

```
kubectl get tasr -o yaml
```

```
apiVersion: trident.netapp.io/v1
items:
- apiVersion: trident.netapp.io/v1
  kind: TridentActionSnapshotRestore
  metadata:
    creationTimestamp: "2023-04-14T00:20:33Z"
    generation: 3
    name: trident-snap
    namespace: trident
    resourceVersion: "3453847"
    uid: <uid>
  spec:
    pvcName: pvc1
    volumeSnapshotName: pvc1-snapshot
  status:
    startTime: "2023-04-14T00:20:34Z"
    completionTime: "2023-04-14T00:20:37Z"
    state: Succeeded
kind: List
metadata:
  resourceVersion: ""
```

- Na maioria dos casos, Trident não tentará novamente a operação automaticamente em caso de falha. Você precisará executar a operação novamente.
- Usuários do Kubernetes sem acesso de administrador podem precisar receber permissão do administrador para criar um TASR CR em seu namespace de aplicação.

Excluir um PV com snapshots associados

Ao excluir um Persistent Volume com snapshots associados, o volume Trident correspondente é atualizado para o estado "Deleting state". Remova os snapshots do volume para excluir o volume Trident.

Implante um controlador de Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de Snapshot de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotclasses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshotcontents.yam
l
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
6.1/client/config/crd/snapshot.storage.k8s.io_volumesnapshots.yaml
```

2. Crie o controlador de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-6.1/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "Instantâneos de volume"
- "VolumeSnapshotClass"

Trabalhar com Snapshots de grupo de volume

Snapshots de grupo de volumes do Kubernetes de Persistent Volumes (PVs) NetApp Trident fornecem a capacidade de criar snapshots de múltiplos volumes (um grupo de snapshots de volumes). Esse snapshot de grupo de volumes representa cópias de múltiplos volumes feitas no mesmo ponto no tempo.

VolumeGroupSnapshot é um recurso beta no Kubernetes com APIs beta. Kubernetes 1.32 é a versão mínima necessária para VolumeGroupSnapshot.

Criar snapshots de grupo de volume

O snapshot de grupo de volume é compatível com os seguintes drivers de armazenamento:

- `ontap-san`driver - somente para os protocolos iSCSI e FC, não para o protocolo NVMe/TCP.
- `ontap-san-economy` - somente para o protocolo iSCSI.
- `ontap-nas`

> O snapshot de grupo de volume não é compatível com sistemas de armazenamento NetApp ASA r2 ou AFX.

Antes de começar

- Certifique-se de que sua versão do Kubernetes seja K8s 1.32 ou superior.
- Você precisa de um controlador de snapshots externo e definições de recursos personalizados (CRDs) para trabalhar com snapshots. Essa é a responsabilidade do orquestrador do Kubernetes (por exemplo: Kubeadm, GKE, OpenShift).

  Se a sua distribuição Kubernetes não incluir o controlador de snapshot externo e os CRDs, consulte Implante um controlador de Snapshot de volume.

  > Não crie um controlador de snapshot se estiver criando snapshots de grupo de volume sob demanda em um ambiente GKE. O GKE usa um controlador de snapshot incorporado e oculto.

- No arquivo YAML do controlador de snapshot, defina o `CSIVolumeGroupSnapshot`feature gate como 'true' para garantir que o snapshot de grupo de volume esteja habilitado.
- Crie as classes de snapshot de grupo de volume necessárias antes de criar um snapshot de grupo de volume.
- Certifique-se de que todos os PVCs/volumes estejam no mesmo SVM para poder criar VolumeGroupSnapshot.

Passos

- Crie um VolumeGroupSnapshotClass antes de criar um VolumeGroupSnapshot. Para obter mais informações, consulte "VolumeGroupSnapshotClass".

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

- Crie PVCs com os rótulos necessários usando as classes de armazenamento existentes ou adicione esses rótulos aos PVCs existentes.

  O exemplo a seguir cria o PVC usando `pvc1-group-snap` como fonte de dados e o rótulo

`consistentGroupSnapshot: groupA`. Defina a chave e o valor do rótulo de acordo com suas necessidades.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: pvc1-group-snap
  labels:
    consistentGroupSnapshot: groupA
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 100Mi
  storageClassName: sc1-1
```

- Crie um VolumeGroupSnapshot com o mesmo rótulo (`consistentGroupSnapshot: groupA` especificado no PVC.

  Este exemplo cria um Snapshot de grupo de volume:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshot
metadata:
  name: "vgs1"
  namespace: trident
spec:
  volumeGroupSnapshotClassName: csi-group-snap-class
  source:
    selector:
      matchLabels:
        consistentGroupSnapshot: groupA
```

Recuperar dados de volume usando um snapshot de grupo

Você pode restaurar Volumes Persistentes individuais usando os snapshots individuais que foram criados como parte do Snapshot do Grupo de Volumes. Você não pode recuperar o Snapshot do Grupo de Volumes como uma unidade.

Use o volume snapshot restore ONTAP CLI para restaurar um volume a um estado registrado em um snapshot anterior.

```
cluster1::*> volume snapshot restore -vserver vs0 -volume vol3 -snapshot
vol3_snap_archive
```

Ao restaurar uma cópia de Snapshot, a configuração do volume existente é sobrescrita. As alterações feitas nos dados do volume após a criação da cópia de Snapshot são perdidas.

Restauração de volume in-place a partir de uma Snapshot

Trident oferece restauração rápida e in-place de volumes a partir de um snapshot usando o `TridentActionSnapshotRestore` (TASR) CR. Este CR funciona como uma ação imperativa do Kubernetes e não persiste após a conclusão da operação.

Para obter mais informações, consulte "Restauração de volume in-place a partir de uma Snapshot".

Excluir um PV com snapshots de grupo associados

Ao excluir um Snapshot de grupo de volume:

- Você pode excluir VolumeGroupSnapshots como um todo, não snapshots individuais do grupo.
- Se PersistentVolumes forem excluídos enquanto existir um snapshot para esse PersistentVolume, Trident moverá esse volume para um estado de "exclusão", pois o snapshot precisa ser removido antes que o volume possa ser removido com segurança.
- Se um clone tiver sido criado usando um snapshot agrupado e, em seguida, o grupo for excluído, uma operação de divisão no clone será iniciada e o grupo não poderá ser excluído até que a divisão seja concluída.

Implante um controlador de Snapshot de volume

Se a sua distribuição Kubernetes não incluir o snapshot controller e os CRDs, você pode implantá-los da seguinte forma.

Passos

1. Criar CRDs de Snapshot de volume.

```
cat snapshot-setup.sh
```

```
#!/bin/bash
# Create volume snapshot CRDs
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotcl
asses.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshotco
ntents.yaml
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-
8.2/client/config/crd/groupsnapshot.storage.k8s.io_volumegroupsnapshots.
yaml
```

2. Crie o controlador de snapshot.

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/rbac-snapshot-controller.yaml
```

```
kubectl apply -f https://raw.githubusercontent.com/kubernetes-
csi/external-snapshotter/release-8.2/deploy/kubernetes/snapshot-
controller/setup-snapshot-controller.yaml
```

Se necessário, abra `deploy/kubernetes/snapshot-controller/rbac-snapshot-controller.yaml` e atualize `namespace` para o seu namespace.

Links relacionados

- "VolumeGroupSnapshotClass"
- "Instantâneos de volume"

Gerencie e monitore Trident

Atualizar Trident

Atualizar Trident

A partir da versão 24.02, Trident segue um ciclo de lançamentos de quatro meses, disponibilizando três versões principais a cada ano civil. Cada nova versão se baseia nas versões anteriores e oferece novos recursos, melhorias de desempenho, correções de bugs e aprimoramentos. Recomendamos que você atualize pelo menos uma vez por ano para aproveitar os novos recursos em Trident.

Considerações antes de atualizar

Ao atualizar para a versão mais recente do Trident, considere o seguinte:

- Deve haver apenas uma instância do Trident instalada em todos os namespaces de um determinado cluster Kubernetes.
- Trident 23.07 e versões posteriores exigem snapshots de volume v1 e não oferecem mais suporte para snapshots alpha ou beta.
- Ao atualizar, é importante que você forneça `parameter.fsType` em `StorageClasses` usado pelo Trident. Você pode excluir e recriar `StorageClasses` sem interromper volumes pré-existentes.
  - Este é um **requisito** para impor "contextos de segurança" para volumes SAN.
  - O diretório sample input contém exemplos, como `storage-class-basic.yaml.templ` e `storage-class-bronze-default.yaml`.
  - Para obter mais informações, consulte "Problemas conhecidos".

Passo 1: selecione uma versão

As versões do Trident seguem uma convenção de nomenclatura baseada em data `YY.MM`, onde "YY" são os dois últimos dígitos do ano e "MM" é o mês. As versões com ponto seguem uma convenção `YY.MM.X`, onde "X" é o nível de patch. Você selecionará a versão para atualizar com base na versão da qual está atualizando.

- Você pode realizar uma atualização direta para qualquer versão de destino que esteja dentro de um intervalo de quatro versões da sua versão instalada. Por exemplo, você pode atualizar diretamente de 24.06 (ou qualquer versão secundária de 24.06) para 25.06.
- Se você estiver atualizando de uma versão fora do período de quatro versões, execute uma atualização em várias etapas. Use as instruções de atualização para a "versão anterior" da qual você está atualizando para atualizar para a versão mais recente que se encaixe no período de quatro versões. Por exemplo, se você estiver executando 23.07 e quiser atualizar para 25.06:
  a. Primeira atualização da 23.07 para a 24.06.
  b. Em seguida, atualize de 24.06 para 25.06.

> Ao atualizar usando o operador Trident na OpenShift Container Platform, você deve atualizar para o Trident 21.01.1 ou posterior. O operador Trident lançado com 21.01.0 contém um problema conhecido que foi corrigido em 21.01.1. Para mais detalhes, consulte o "detalhes do problema em GitHub".

Etapa 2: determine o método de instalação original

Para determinar qual versão você usou para instalar originalmente o Trident:

1. Utilize `kubectl get pods -n trident` para examinar os pods.
   - Se não houver um pod de operador, Trident foi instalado usando `tridentctl`.
   - Caso exista um pod de operador, Trident foi instalado usando o Trident operator, seja manualmente ou usando Helm.
2. Se houver um pod de operador, use `kubectl describe torc` para determinar se Trident foi instalado usando Helm.
   - Se houver uma etiqueta Helm, Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, Trident foi instalado manualmente usando o operador Trident.

Etapa 3: selecione um método de atualização

Geralmente, você deve atualizar usando o mesmo método que usou para a instalação inicial, porém você pode "alternar entre métodos de instalação". Existem duas opções para atualizar Trident.

- "Faça o upgrade usando o operador Trident"

> Sugerimos que você revise "Entenda o fluxo de trabalho de atualização do operador" antes de fazer o upgrade com o operador.

*

Atualize com o operador

Entenda o fluxo de trabalho de atualização do operador

Antes de usar o operador Trident para atualizar o Trident, você deve entender os processos em segundo plano que ocorrem durante a atualização. Isso inclui alterações no controlador Trident, no Pod do controlador e nos Pods dos nós, e no DaemonSet dos nós que permitem atualizações contínuas.

Gerenciamento de atualização do operador Trident

Uma das muitas "benefícios de usar o operador Trident" maneiras de instalar e atualizar Trident é o gerenciamento automático de objetos do Trident e do Kubernetes sem interromper os volumes montados existentes. Dessa forma, Trident pode suportar atualizações sem inatividade, ou "*atualizações contínuas*". Em particular, o operador do Trident se comunica com o cluster Kubernetes para:

- Exclua e recrie a implantação do Trident Controller e o DaemonSet do nó.
- Substitua o Trident Controller Pod e os Trident Node Pods por novas versões.
  - Se um nó não for atualizado, isso não impede que os nós restantes sejam atualizados.
  - Somente nós com um Trident Node Pod em execução podem montar volumes.

> Para obter mais informações sobre a arquitetura do Trident no cluster Kubernetes, consulte "Arquitetura do Trident".

Fluxo de trabalho de atualização do operador

Ao iniciar uma atualização usando o Trident operator:

1. O **operador Trident**:
   a. Detecta a versão atualmente instalada do Trident (versão *n*).
   b. Atualiza todos os objetos do Kubernetes, incluindo CRDs, RBAC e Trident SVC.
   c. Exclui a implantação do Trident Controller para a versão *n*.
   d. Cria a implantação do Trident Controller para a versão *n+1*.
2. **Kubernetes** cria o Pod do Controlador Trident para *n+1*.
3. O **operador Trident**:
   a. Exclui o Trident Node DaemonSet para *n*. O operador não espera pelo término do Pod do nó.
   b. Cria o Daemonset Trident Node para *n+1*.
4. **Kubernetes** cria Trident Node Pods em nós que não estão executando Trident Node Pod *n*. Isso garante que nunca haja mais de um Trident Node Pod, de qualquer versão, em um nó.

Atualize uma instalação do Trident usando o Trident operator ou o Helm

Você pode atualizar o Trident usando o operador Trident, seja manualmente ou usando o Helm. Você pode atualizar de uma instalação do operador Trident para outra instalação do operador Trident ou atualizar de uma `tridentctl` instalação para uma versão do operador Trident. Revise "Selecione um método de atualização" antes de atualizar uma instalação do operador Trident.

Atualize uma instalação manual

Você pode atualizar de uma instalação do operador Trident com escopo de cluster para outra instalação do operador Trident com escopo de cluster. Todas as versões do Trident usam um operador com escopo de cluster.

> Para atualizar a partir do Trident instalado usando o operador com escopo de namespace (versões 20.07 a 20.10), use as instruções de atualização para "sua versão instalada" do Trident.

Sobre esta tarefa

Trident fornece um arquivo de pacote que você pode usar para instalar o operador e criar objetos associados para sua versão do Kubernetes.

- Para clusters executando Kubernetes 1.24, use "bundle_pre_1_25.yaml".
- Para clusters executando Kubernetes 1.25 ou posterior, use "bundle_post_1_25.yaml".

Antes de começar

Certifique-se de estar usando um cluster Kubernetes em execução "uma versão do Kubernetes compatível".

Passos

1. Verifique sua versão do Trident:

```
./tridentctl -n trident version
```

2. Atualize o `operator.yaml`, `tridentorchestrator_cr.yaml`e `post_1_25_bundle.yaml`com o registro e os caminhos de imagem para a versão para a qual você está atualizando (por exemplo, 25.06) e o segredo correto.
3. Exclua o operador Trident que foi usado para instalar a instância Trident atual. Por exemplo, se você estiver atualizando da 25.02, execute o seguinte comando:

```
kubectl delete -f 25.02.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

4. Se você personalizou sua instalação inicial usando `TridentOrchestrator` atributos, pode editar o objeto `TridentOrchestrator` para modificar os parâmetros de instalação. Isso pode incluir alterações feitas para especificar registros de imagens espelhados do Trident e do CSI para o modo offline, ativar logs de depuração ou especificar segredos de extração de imagens.
5. Instale Trident usando o arquivo YAML de bundle correto para o seu ambiente, onde *<bundle.yaml>* é `bundle_pre_1_25.yaml` ou `bundle_post_1_25.yaml` com base na sua versão do Kubernetes. Por exemplo, se você estiver instalando Trident 25.06.0, execute o seguinte comando:

```
kubectl create -f 25.06.0/trident-installer/deploy/<bundle.yaml> -n
trident
```

6. Edite o trident torc para incluir a imagem 25.06.0.

Atualize uma instalação Helm

Você pode atualizar uma instalação do Trident.

> Ao atualizar um cluster Kubernetes da versão 1.24 para a 1.25 ou posterior que tenha Trident instalado, você deve atualizar o arquivo values.yaml para definir `excludePodSecurityPolicy` para `true` ou adicionar `--set excludePodSecurityPolicy=true` ao comando `helm upgrade` antes de poder atualizar o cluster.

Se você já atualizou seu cluster Kubernetes da versão 1.24 para a 1.25 sem atualizar o Trident helm, a atualização do helm falhará. Para que a atualização do helm seja concluída, execute estas etapas como pré-requisitos:

1. Instale o plugin helm-mapkubeapis de https://github.com/helm/helm-mapkubeapis.
2. Execute um teste de execução para o release do Trident no namespace onde Trident está instalado. Isso lista os recursos que serão limpos.

```
helm mapkubeapis --dry-run trident  --namespace trident
```

3. Execute uma execução completa com o helm para fazer a limpeza.

```
helm mapkubeapis trident  --namespace trident
```

Passos

1. Se você "instalou o Trident usando Helm", pode usar `helm upgrade trident netapp-trident/trident-operator --version 100.2506.0` para atualizar em uma única etapa. Se você não adicionou o repositório Helm ou não consegue usá-lo para atualizar:
   a. Baixe a versão mais recente do Trident em "a seção *Ativos* em GitHub".
   b. Utilize o `helm upgrade` comando onde `trident-operator-25.10.0.tgz` reflete a versão para a qual você deseja atualizar.

   ```
   helm upgrade <name> trident-operator-25.10.0.tgz
   ```

   > Se você definir opções personalizadas durante a instalação inicial (como especificar registros privados e espelhados para as imagens do Trident e CSI), acrescente o `helm upgrade` comando usando `--set` para garantir que essas opções sejam incluídas no comando de upgrade, caso contrário, os valores serão redefinidos para o padrão.

2. Execute `helm list` para verificar se o gráfico e a versão do aplicativo foram atualizados. Execute `tridentctl logs` para revisar quaisquer mensagens de depuração.

Atualize de uma `tridentctl` instalação para Trident operator

Você pode atualizar para a versão mais recente do operador Trident a partir de uma `tridentctl` instalação. Os backends e PVCs existentes estarão automaticamente disponíveis.

> Antes de alternar entre os métodos de instalação, revise "Alternando entre métodos de instalação".

Passos

1. Baixe a versão mais recente do Trident.

```
# Download the release required [25.10.0]
mkdir 25.10.0
cd 25.10.0
wget
https://github.com/NetApp/trident/releases/download/v25.10.0/trident-installer-25.10.0.tar.gz
tar -xf trident-installer-25.10.0.tar.gz
cd trident-installer
```

2. Crie o `tridentorchestrator`CRD a partir do manifesto.

```
kubectl create -f
deploy/crds/trident.netapp.io_tridentorchestrators_crd_post1.16.yaml
```

3. Implante o operador com escopo de cluster no mesmo namespace.

```
kubectl create -f deploy/<bundle-name.yaml>

serviceaccount/trident-operator created
clusterrole.rbac.authorization.k8s.io/trident-operator created
clusterrolebinding.rbac.authorization.k8s.io/trident-operator created
deployment.apps/trident-operator created
podsecuritypolicy.policy/tridentoperatorpods created

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-controller-79df798bdc-m79dc   6/6     Running   0          150d
trident-node-linux-xrst8              2/2     Running   0          150d
trident-operator-5574dbbc68-nthjv     1/1     Running   0          1m30s
```

4. Crie um `TridentOrchestrator` CR para instalação do Trident.

```
cat deploy/crds/tridentorchestrator_cr.yaml
apiVersion: trident.netapp.io/v1
kind: TridentOrchestrator
metadata:
  name: trident
spec:
  debug: true
  namespace: trident

kubectl create -f deploy/crds/tridentorchestrator_cr.yaml

#Examine the pods in the Trident namespace
NAME                                READY   STATUS    RESTARTS   AGE
trident-csi-79df798bdc-m79dc        6/6     Running   0          1m
trident-csi-xrst8                   2/2     Running   0          1m
trident-operator-5574dbbc68-nthjv   1/1     Running   0          5m41s
```

5. Confirme se Trident foi atualizado para a versão pretendida.

```
kubectl describe torc trident | grep Message -A 3

Message:                Trident installed
Namespace:              trident
Status:                 Installed
Version:                v25.10.0
```

Atualizar com tridentctl

Você pode facilmente atualizar uma instalação existente do Trident usando `tridentctl`.

Sobre esta tarefa

Desinstalar e reinstalar Trident funciona como uma atualização. Ao desinstalar Trident, o Persistent Volume Claim (PVC) e o Persistent Volume (PV) usados pela implantação do Trident não são excluídos. Os PVs que já foram provisionados permanecerão disponíveis enquanto Trident estiver offline, e Trident provisionará volumes para quaisquer PVCs que forem criados nesse meio tempo, após voltar a ficar online.

Antes de começar

Analise "Selecione um método de atualização" antes de atualizar usando `tridentctl`.

Passos

1. Execute o comando de desinstalação em `tridentctl` para remover todos os recursos associados ao Trident, exceto os CRDs e objetos relacionados.

```
./tridentctl uninstall -n <namespace>
```

2. Reinstale Trident. Consulte "Instale Trident usando tridentctl".

> Não interrompa o processo de atualização. Certifique-se de que o instalador seja executado até o fim.

Gerencie Trident usando tridentctl

O "Pacote de instalação Trident" inclui o `tridentctl` utilitário de linha de comando para fornecer acesso simples ao Trident. Usuários do Kubernetes com privilégios suficientes podem usá-lo para instalar Trident ou gerenciar o namespace que contém o pod do Trident.

Comandos e flags globais

Você pode executar `tridentctl help` para obter uma lista de comandos disponíveis para `tridentctl` ou adicionar a flag `--help` a qualquer comando para obter uma lista de opções e flags para esse comando específico.

```
tridentctl [command] [--optional-flag]
```

O utilitário Trident `tridentctl` suporta os seguintes comandos e flags globais.

Comandos

`create`
Adicionar um recurso ao Trident.

`delete`
Remover um ou mais recursos do Trident.

`get`
Obtenha um ou mais recursos do Trident.

`help`
Ajuda sobre qualquer comando.

`images`
Imprima uma tabela das imagens de contêiner que o Trident precisa.

`import`
Importe um recurso existente para Trident.

`install`
Instale Trident.

`logs`
Imprima os logs do Trident.

`send`
Envie um recurso do Trident.

`uninstall`
Desinstale o Trident.

`update`
Modificar um recurso no Trident.

`update backend state`
Suspender temporariamente as operações de backend.

`upgrade`
Atualize um recurso no Trident.

`version`
Imprima a versão do Trident.

Bandeiras globais

`-d, --debug`
Saída de depuração.

`-h, --help`
Ajuda para `tridentctl`.

`-k, --kubeconfig string`
Especifique o `KUBECONFIG` caminho para executar comandos localmente ou de um cluster Kubernetes para outro.

Alternativamente, você pode exportar a `KUBECONFIG` variável para apontar para um cluster Kubernetes específico e emitir `tridentctl` comandos para esse cluster.

`-n, --namespace string`
Namespace da implantação do Trident.

`-o, --output string`
Formato de saída. Um dos seguintes: json|yaml|name|wide|ps (padrão).

`-s, --server string`
Endereço/porta da interface API REST do Trident.

A interface REST do Trident pode ser configurada para escutar e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

Opções e parâmetros de comando

criar

Use o `create` comando para adicionar um recurso ao Trident.

```
tridentctl create [option]
```

Opções

`backend`: Adicione um backend ao Trident.

excluir

Use o comando `delete` para remover um ou mais recursos do Trident.

```
tridentctl delete [option]
```

Opções

`backend`: excluir um ou mais backends de armazenamento do Trident.
`snapshot`: excluir um ou mais snapshots de volume do Trident.
`storageclass`: excluir uma ou mais classes de armazenamento do Trident.

`volume`: excluir um ou mais volumes de armazenamento do Trident.

get

Use o comando `get` para obter um ou mais recursos do Trident.

```
tridentctl get [option]
```

Opções

`backend`: obtenha um ou mais backends de armazenamento do Trident.
`snapshot`: obtenha um ou mais snapshots do Trident.
`storageclass`: obtenha uma ou mais storage classes do Trident.
`volume`: obtenha um ou mais volumes do Trident.

Bandeiras

`-h`, `--help`: ajuda para volumes.
`--parentOfSubordinate string`: limitar a consulta ao volume de origem subordinado.
`--subordinateOf string`: limitar a consulta aos subordinados do volume.

imagens

Use `images` flags para imprimir uma tabela com as imagens dos contêineres que o Trident precisa.

```
tridentctl images [flags]
```

Bandeiras

`-h`, `--help`: ajuda para imagens.
`-v`, `--k8s-version string`: versão semântica do cluster Kubernetes.

importar volume

Use o `import volume` comando para importar um volume existente para Trident.

```
tridentctl import volume <backendName> <volumeName> [flags]
```

Aliases

`volume`, `v`

Bandeiras

`-f`, `--filename string`: Caminho para o arquivo PVC em YAML ou JSON.
`-h`, `--help`: Ajuda para volume.
`--no-manage`: Criar somente PV/PVC. Não assumir gerenciamento de ciclo de vida do volume.

instalar

Use os `install` flags para instalar Trident.

```
tridentctl install [flags]
```

Bandeiras

`--autosupport-image string`: A imagem do contêiner para Autosupport Telemetry (padrão "netapp/trident autosupport:<current-version>").
`--autosupport-proxy string`: O endereço/porta de um proxy para envio de Autosupport Telemetry.
`--enable-node-prep`: Tentar instalar os pacotes necessários nos nós.
`--generate-custom-yaml`: Gerar arquivos YAML sem instalar nada.
`-h, --help`: Ajuda para instalação.
`--http-request-timeout`: Substituir o tempo limite de solicitação HTTP para a API REST do Trident (padrão 1m30s).
`--image-registry string`: O endereço/porta de um registro de imagens interno.
`--k8s-timeout duration`: O tempo limite para todas as operações do Kubernetes (padrão 3m0s).
`--kubelet-dir string`: O local do host do estado interno do kubelet (padrão "/var/lib/kubelet").
`--log-format string`: O formato de log do Trident (text, json) (padrão "text").
`--node-prep`: Permite que o Trident prepare os nós do cluster Kubernetes para gerenciar volumes usando o protocolo de storage de dados especificado. **Atualmente, `iscsi` é o único valor suportado. A partir do OpenShift 4.19, a versão mínima do Trident compatível com este recurso é 25.06.1.**
`--pv string`: O nome do PV legado usado pelo Trident; certifique-se de que ele não exista (padrão "trident").
`--pvc string`: O nome do PVC legado usado pelo Trident; certifique-se de que ele não exista (padrão "trident").
`--silence-autosupport`: Não enviar pacotes autosupport para NetApp automaticamente (padrão true).
`--silent`: Desativar a maior parte da saída durante a instalação.
`--trident-image string`: A imagem do Trident a ser instalada.
`--k8s-api-qps`: O limite de consultas por segundo (QPS) para solicitações da API do Kubernetes (padrão 100; opcional).
`--use-custom-yaml`: Usar quaisquer arquivos YAML existentes no diretório de setup.
`--use-ipv6`: Usar IPv6 para a comunicação do Trident.

registros

Use `logs`flags para imprimir os logs do Trident.

```
tridentctl logs [flags]
```

Bandeiras

`-a, --archive`: Crie um arquivo de suporte com todos os logs, a menos que especificado de outra forma.
`-h, --help`: Ajuda para logs.
`-l, --log string`: Log do Trident a ser exibido. Um de trident|auto|trident-operator|all (padrão "auto").
`--node string`: O nome do nó do Kubernetes do qual coletar os logs do pod do nó.
`-p, --previous`: Obtenha os logs da instância anterior do contêiner, se existir.
`--sidecars`: Obtenha os logs dos contêineres sidecar.

enviar

Use o comando `send` para enviar um recurso do Trident.

```
tridentctl send [option]
```

Opções

`autosupport`: Envie um arquivo de Autosupport para NetApp.

desinstalar

Use `uninstall`parâmetros para desinstalar Trident.

```
tridentctl uninstall [flags]
```

Bandeiras

`-h, --help`: Ajuda para desinstalar.
`--silent`: Desativar a maior parte da saída durante a desinstalação.

atualizar

Use o `update` comando para modificar um recurso no Trident.

```
tridentctl update [option]
```

Opções

`backend`: Atualizar um backend no Trident.

atualizar estado do backend

Use o comando `update backend state` para suspender ou retomar as operações de backend.

```
tridentctl update backend state <backend-name> [flag]
```

Pontos a considerar

- Se um backend for criado usando um TridentBackendConfig (tbc), o backend não poderá ser atualizado usando um arquivo `backend.json`.
- Se o `userState` tiver sido definido em um tbc, ele não poderá ser modificado usando o comando `tridentctl update backend state <backend-name> --user-state suspended/normal`.
- Para recuperar a capacidade de definir o `userState` via tridentctl após ter sido definido via tbc, o campo `userState` deve ser removido do tbc. Isso pode ser feito usando o comando `kubectl edit tbc`. Após o campo `userState` ser removido, você pode usar o comando `tridentctl update backend state` para alterar o `userState` de um backend.
- Use o `tridentctl update backend state` para alterar o `userState`. Você também pode atualizar o `userState` usando `TridentBackendConfig` ou `backend.json` arquivo; isso aciona uma reinicialização completa do backend e pode ser demorado.

  #### Bandeiras

  `-h, --help`: Ajuda para o estado do backend.
  `--user-state`: Defina como `suspended` para pausar as operações do backend. Defina como `normal` para retomar as operações do backend. Quando definido como `suspended`:

- `AddVolume` e `Import Volume` estão em pausa.
- `CloneVolume`, `ResizeVolume`, `PublishVolume`, `UnPublishVolume`, `CreateSnapshot`, `GetSnapshot`, `RestoreSnapshot`, `DeleteSnapshot`, `RemoveVolume`, `GetVolumeExternal`, `ReconcileNodeAccess` permanecem disponíveis.

Você também pode atualizar o estado do backend usando `userState` campo no arquivo de configuração do backend `TridentBackendConfig` ou `backend.json`. Para mais informações, consulte "Opções para

gerenciamento de backends" e "Realize o gerenciamento de backend com kubectl".

Exemplo:

JSON

Siga estes passos para atualizar o `userState` usando o arquivo `backend.json`:

1. Edite o `backend.json` arquivo para incluir o `userState` campo com o valor definido como 'suspended'.
2. Atualize o backend usando o `tridentctl update backend` comando e o caminho para o arquivo `backend.json` atualizado.

   Exemplo: `tridentctl update backend -f /<path to backend JSON file>/backend.json -n trident`

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "<redacted>",
  "svm": "nas-svm",
  "backendName": "customBackend",
  "username": "<redacted>",
  "password": "<redacted>",
  "userState": "suspended"
}
```

YAML

Você pode editar o tbc depois que ele for aplicado usando o `kubectl edit <tbc-name> -n <namespace>` comando. O exemplo a seguir atualiza o estado do backend para suspender usando a `userState: suspended` opção:

```
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  backendName: customBackend
  storageDriverName: ontap-nas
  managementLIF: <redacted>
  svm: nas-svm
  userState: suspended
  credentials:
    name: backend-tbc-ontap-nas-secret
```

versão

Use `version` flags para imprimir a versão de `tridentctl` e o serviço Trident em execução.

```
tridentctl version [flags]
```

Bandeiras

`--client`: somente versão cliente (não requer servidor).
`-h, --help`: ajuda para versão.

Suporte a plugin

O Tridentctl suporta plugins de forma semelhante ao kubectl. O Tridentctl detecta um plugin se o nome do arquivo binário do plugin seguir o esquema "tridentctl-<plugin>" e o binário estiver localizado em uma pasta listada na variável de ambiente PATH. Todos os plugins detectados são listados na seção de plugins da ajuda do tridentctl. Opcionalmente, você também pode limitar a busca especificando uma pasta de plugin na variável de ambiente TRIDENTCTL_PLUGIN_PATH (Exemplo: `TRIDENTCTL_PLUGIN_PATH=~/tridentctl-plugins/`). Se a variável for usada, o tridentctl buscará apenas na pasta especificada.

Monitorar Trident

Trident fornece um conjunto de endpoints de métricas do Prometheus que você pode usar para monitorar o desempenho do Trident.

Visão geral

As métricas fornecidas pela Trident permitem que você faça o seguinte:

- Monitore a saúde e a configuração do Trident. Você pode examinar o sucesso das operações e se ele consegue se comunicar com os backends conforme o esperado.
- Analise as informações de utilização do backend e entenda quantos volumes estão provisionados em um backend e a quantidade de espaço consumido, e assim por diante.
- Mantenha um mapeamento da quantidade de volumes provisionados nos backends disponíveis.
- Monitore o desempenho. Você pode verificar quanto tempo Trident leva para se comunicar com os backends e executar operações.

> Por padrão, as métricas do Trident são expostas na porta de destino `8001` no endpoint `/metrics`. Essas métricas são **ativadas por padrão** quando Trident é instalado. Você pode configurar para consumir as métricas do Trident via HTTPS na porta `8444` também.

O que você vai precisar

- Um cluster do Kubernetes com Trident instalado.
- Uma instância do Prometheus. Isso pode ser um "implantação do Prometheus em contêiner" ou você pode optar por executar o Prometheus como um "aplicativo nativo".

Etapa 1: defina um alvo do Prometheus

Você deve definir um alvo Prometheus para coletar as métricas e obter informações sobre os backends que o Trident gerencia, os volumes que ele cria e assim por diante. Veja "Documentação do Prometheus Operator".

Passo 2: criar um Prometheus ServiceMonitor

Para consumir as métricas do Trident, você deve criar um Prometheus ServiceMonitor que monitore o `trident-csi` serviço e escute na `metrics` porta. Um exemplo de ServiceMonitor é o seguinte:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - port: metrics
      interval: 15s
```

Esta definição de ServiceMonitor recupera as métricas retornadas pelo `trident-csi` serviço e procura especificamente pelo endpoint `metrics` do serviço. Como resultado, o Prometheus agora está configurado para entender as métricas do Trident.

Além das métricas disponíveis diretamente no Trident, kubelet expõe muitas `kubelet_volume_*` métricas por meio de seu próprio endpoint de métricas. Kubelet pode fornecer informações sobre os volumes que estão anexados, e pods e outras operações internas que gerencia. Consulte "aqui".

Consuma métricas do Trident por HTTPS

Para consumir métricas do Trident via HTTPS (porta 8444), você deve modificar a definição do ServiceMonitor para incluir a configuração TLS. Você também precisa copiar o `trident-csi` segredo do `trident` namespace para o namespace onde o Prometheus está sendo executado. Você pode fazer isso usando o seguinte comando:

```
kubectl get secret trident-csi -n trident -o yaml | sed 's/namespace:
trident/namespace: monitoring/' | kubectl apply -f -
```

Um exemplo de ServiceMonitor para métricas HTTPS é o seguinte:

```
apiVersion: monitoring.coreos.com/v1
kind: ServiceMonitor
metadata:
  name: trident-sm
  namespace: monitoring
  labels:
    release: prom-operator
spec:
  jobLabel: trident
  selector:
    matchLabels:
      app: controller.csi.trident.netapp.io
  namespaceSelector:
    matchNames:
      - trident
  endpoints:
    - interval: 15s
      path: /metrics
      port: https-metrics
      scheme: https
      tlsConfig:
        ca:
          secret:
            key: caCert
            name: trident-csi
        cert:
          secret:
            key: clientCert
            name: trident-csi
        keySecret:
          key: clientKey
          name: trident-csi
        serverName: trident-csi
```

Trident oferece suporte a métricas HTTPS em todos os métodos de instalação: tridentctl, Helm chart e Operator:

- Se você estiver usando o `tridentctl install` comando, pode passar a `--https-metrics` flag para habilitar métricas HTTPS.
- Se você estiver usando o gráfico Helm, você pode definir o parâmetro `httpsMetrics` para habilitar métricas HTTPS.
- Se você estiver usando arquivos YAML, pode adicionar a `--https_metrics` flag ao `trident-main` contêiner no `trident-deployment.yaml` arquivo.

Etapa 3: Consultar métricas do Trident com PromQL

PromQL é bom para criar expressões que retornam séries temporais ou dados tabulares.

Aqui estão algumas consultas PromQL que você pode usar:

Obtenha informações de integridade do Trident

- **Porcentagem de respostas HTTP 2XX do Trident**

```
(sum (trident_rest_ops_seconds_total_count{status_code=~"2.."} OR on()
vector(0)) / sum (trident_rest_ops_seconds_total_count)) * 100
```

- **Percentual de respostas REST do Trident via código de status**

```
(sum (trident_rest_ops_seconds_total_count) by (status_code)  / scalar
(sum (trident_rest_ops_seconds_total_count))) * 100
```

- **Duração média em ms das operações realizadas pelo Trident**

```
sum by (operation)
(trident_operation_duration_milliseconds_sum{success="true"}) / sum by
(operation)
(trident_operation_duration_milliseconds_count{success="true"})
```

Obtenha informações de uso do Trident

- **Tamanho médio do volume**

```
trident_volume_allocated_bytes/trident_volume_count
```

- **Espaço total de volume provisionado por cada backend**

```
sum (trident_volume_allocated_bytes) by (backend_uuid)
```

Obtenha o uso de volume individual

Isso só é habilitado se as métricas do kubelet também forem coletadas.

- **Percentual de espaço utilizado para cada volume**

```
kubelet_volume_stats_used_bytes / kubelet_volume_stats_capacity_bytes *
100
```

Saiba mais sobre a telemetria Trident AutoSupport

Por padrão, Trident envia métricas do Prometheus e informações básicas de backend para a NetApp em uma cadência diária.

- Para impedir que o Trident envie métricas do Prometheus e informações básicas de backend para NetApp, passe a `--silence-autosupport` flag durante a instalação do Trident.
- Trident também pode enviar logs de contêiner para o suporte da NetApp sob demanda via `tridentctl send autosupport`. Você precisará acionar o Trident para fazer upload dos logs. Antes de enviar os logs, você deve aceitar a NetApphttps://www.netapp.com/company/legal/privacy-policy/["política de privacidade"^].
- A menos que especificado, Trident busca os logs das últimas 24 horas.
- Você pode especificar o período de retenção de logs com a `--since` flag. Por exemplo: `tridentctl send autosupport --since=1h`. Essas informações são coletadas e enviadas por meio de um `trident-autosupport` contêiner que é instalado junto com Trident. Você pode obter a imagem do contêiner em "Trident AutoSupport".
- Trident AutoSupport não coleta nem transmite Informações de Identificação Pessoal (PII) ou Informações Pessoais. Vem com uma "Contrato de licença do usuário final" que não se aplica à própria imagem do contêiner Trident. Você pode saber mais sobre o compromisso da NetApp com a segurança e a confiança dos dados "aqui".

Um exemplo de carga enviada pelo Trident é semelhante a esta:

```
---
items:
  - backendUUID: ff3852e1-18a5-4df4-b2d3-f59f829627ed
    protocol: file
    config:
      version: 1
      storageDriverName: ontap-nas
      debug: false
      debugTraceFlags: null
      disableDelete: false
      serialNumbers:
        - nwkvzfanek_SN
      limitVolumeSize: ""
    state: online
    online: true
```

- As mensagens do AutoSupport são enviadas para o endpoint do AutoSupport da NetApp. Se você estiver usando um registro privado para armazenar imagens de contêiner, poderá usar a flag `--image -registry`.

- Você também pode configurar URLs de proxy gerando os arquivos YAML de instalação. Isso pode ser feito usando `tridentctl install --generate-custom-yaml` para criar os arquivos YAML e adicionando o `--proxy-url` argumento para o contêiner `trident-autosupport` em `trident-deployment.yaml`.

Desativar métricas do Trident

Para **desativar** a geração de relatórios de métricas, você deve gerar arquivos YAML personalizados (usando a `--generate-custom-yaml` flag) e editá-los para remover a `--metrics` flag da invocação do contêiner `trident-main`.

Desinstalar Trident

Você deve usar o mesmo método para desinstalar Trident que usou para instalar Trident.

Sobre esta tarefa

- Se você precisar corrigir erros observados após uma atualização, problemas de dependência ou uma atualização incompleta ou malsucedida, você deve desinstalar Trident e reinstalar a versão anterior usando as instruções específicas para essa "versão". Esta é a única maneira recomendada de *fazer downgrade* para uma versão anterior.
- Para facilitar a atualização e reinstalação, desinstalar Trident não remove os CRDs ou objetos relacionados criados por Trident. Se precisar remover completamente Trident e todos os seus dados, consulte "Remova completamente Trident e os CRDs".

Antes de começar

Se você estiver desativando clusters Kubernetes, deverá excluir todos os aplicativos que utilizam volumes criados pelo Trident antes da desinstalação. Isso garante que os PVCs sejam despublicados nos nós do Kubernetes antes de serem excluídos.

Determine o método de instalação original

Você deve usar o mesmo método para desinstalar Trident que usou para instalá-lo. Antes de desinstalar, verifique qual versão você usou para instalar originalmente o Trident.

1. Utilize `kubectl get pods -n trident` para examinar os pods.
   - Se não houver um pod de operador, Trident foi instalado usando `tridentctl`.
   - Caso exista um pod de operador, Trident foi instalado usando o Trident operator, seja manualmente ou usando Helm.
2. Se houver um pod de operador, use `kubectl describe tproc trident` para determinar se Trident foi instalado usando Helm.
   - Se houver uma etiqueta Helm, Trident foi instalado usando Helm.
   - Se não houver nenhuma etiqueta Helm, Trident foi instalado manualmente usando o operador Trident.

Desinstalar uma instalação do Trident

Você pode desinstalar uma instalação do operador Trident manualmente ou usando Helm.

Desinstalar instalação manual

Se você instalou Trident usando o operador, pode desinstalá-lo fazendo uma das seguintes ações:

1. **Edite `TridentOrchestrator` o CR e defina o sinalizador de desinstalação**:

   ```
   kubectl patch torc <trident-orchestrator-name> --type=merge -p
   '{"spec":{"uninstall":true}}'
   ```

   Quando a `uninstall` flag está definida como `true`, o operador Trident desinstala Trident, mas não remove o TridentOrchestrator em si. Você deve limpar o TridentOrchestrator e criar um novo se quiser instalar Trident novamente.

2. **Excluir `TridentOrchestrator`**: Ao remover o `TridentOrchestrator` CR que foi usado para implantar Trident, você instrui o operador a desinstalar Trident. O operador processa a remoção de `TridentOrchestrator` e procede à remoção da implantação e do daemonset do Trident, excluindo os pods do Trident que havia criado como parte da instalação.

   ```
   kubectl delete -f deploy/<bundle.yaml> -n <namespace>
   ```

Desinstalar instalação Helm

Se você instalou Trident usando Helm, pode desinstalá-lo usando `helm uninstall`.

```
#List the Helm release corresponding to the Trident install.
helm ls -n trident
NAME          NAMESPACE       REVISION        UPDATED
STATUS          CHART                           APP VERSION
trident       trident         1               2021-04-20
00:26:42.417764794 +0000 UTC deployed        trident-operator-21.07.1
21.07.1

#Uninstall Helm release to remove Trident
helm uninstall trident -n trident
release "trident" uninstalled
```

Desinstalar uma `tridentctl` instalação

Use o `uninstall` comando em `tridentctl` para remover todos os recursos associados ao Trident, exceto os CRDs e objetos relacionados:

```
./tridentctl uninstall -n <namespace>
```

Trident for Docker

Pré-requisitos para implantação

Você precisa instalar e configurar os pré-requisitos de protocolo necessários em seu host antes de poder implantar Trident.

Verifique os requisitos

- Verifique se sua implantação atende a todos os "requisitos".
- Verifique se você tem uma versão compatível do Docker instalada. Se a sua versão do Docker estiver desatualizada, "instalar ou atualizar".

```
docker --version
```

- Verifique se os pré-requisitos do protocolo estão instalados e configurados no seu host.

Ferramentas NFS

Instale as ferramentas NFS usando os comandos para o seu sistema operacional.

RHEL 8+

```
sudo yum install -y nfs-utils
```

Ubuntu

```
sudo apt-get install -y nfs-common
```

Reinicie os nós de trabalho após instalar as ferramentas NFS para evitar falhas ao conectar volumes aos contêineres.

Ferramentas iSCSI

Instale as ferramentas iSCSI usando os comandos para o seu sistema operacional.

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi iscsi-initiator-utils sg3_utils device-
mapper-multipath
```

2. Verifique se a versão do iscsi-initiator-utils é 6.2.0.874-2.el7 ou posterior:

```
rpm -q iscsi-initiator-utils
```

3. Defina a digitalização como manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `iscsid` e `multipathd` estejam em execução:

```
sudo systemctl enable --now iscsid multipathd
```

6. Ativar e iniciar `iscsi`:

```
sudo systemctl enable --now iscsi
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y open-iscsi lsscsi sg3-utils multipath-tools
scsitools
```

2. Verifique se a versão do open-iscsi é 2.0.874-5ubuntu2.10 ou posterior (para bionic) ou 2.0.874-7.1ubuntu6.1 ou posterior (para focal):

```
dpkg -l open-iscsi
```

3. Defina a digitalização como manual:

```
sudo sed -i 's/^\(node.session.scan\).*/\1 = manual/'
/etc/iscsi/iscsid.conf
```

4. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

5. Certifique-se de que `open-iscsi` e `multipath-tools` estejam habilitados e em execução:

```
sudo systemctl status multipath-tools
sudo systemctl enable --now open-iscsi.service
sudo systemctl status open-iscsi
```

Ferramentas NVMe

Instale as ferramentas NVMe usando os comandos para o seu sistema operacional.

- NVMe requer RHEL 9 ou posterior.
- Se a versão do kernel do seu nó Kubernetes for muito antiga ou se o pacote NVMe não estiver disponível para a sua versão do kernel, talvez seja necessário atualizar a versão do kernel do seu nó para uma com o pacote NVMe.

RHEL 9

```
sudo yum install nvme-cli
sudo yum install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ubuntu

```
sudo apt install nvme-cli
sudo apt -y install linux-modules-extra-$(uname -r)
sudo modprobe nvme-tcp
```

Ferramentas FC

Instale as ferramentas FC usando os comandos para o seu sistema operacional.

- Ao usar nós de trabalho que executam RHEL/Red Hat Enterprise Linux CoreOS (RHCOS) com FC PVs, especifique o `discard` mountOption na StorageClass para realizar exigência de espaço inline. Consulte "Documentação Red Hat".

RHEL 8+

1. Instale os seguintes pacotes do sistema:

```
sudo yum install -y lsscsi device-mapper-multipath
```

2. Ative multipathing:

```
sudo mpathconf --enable --with_multipathd y --find_multipaths n
```

> Certifique-se de que `etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipathd` está em execução:

```
sudo systemctl enable --now multipathd
```

Ubuntu

1. Instale os seguintes pacotes do sistema:

```
sudo apt-get install -y lsscsi sg3-utils multipath-tools scsitools
```

2. Ative multipathing:

```
sudo tee /etc/multipath.conf <<-EOF
defaults {
    user_friendly_names yes
    find_multipaths no
}
EOF
sudo systemctl enable --now multipath-tools.service
sudo service multipath-tools restart
```

> Certifique-se de que `etc/multipath.conf` contenha `find_multipaths no` sob `defaults`.

3. Certifique-se de que `multipath-tools` está ativado e em execução:

```
sudo systemctl status multipath-tools
```

Implantar Trident

Trident for Docker oferece integração direta com o ecossistema Docker para plataformas de storage NetApp. Ele suporta o provisionamento e o gerenciamento de recursos de storage da plataforma de storage para hosts Docker, com uma estrutura para adicionar plataformas adicionais no futuro.

Várias instâncias do Trident podem ser executadas simultaneamente no mesmo host. Isso permite conexões simultâneas a múltiplos sistemas de storage e tipos de storage, com a possibilidade de personalizar o storage usado para os volumes do Docker.

O que você vai precisar

Consulte o "Pré-requisitos para implantação". Depois de garantir que os pré-requisitos sejam atendidos, você está pronto para implantar Trident.

Método de plugin gerenciado pelo Docker (versão 1.13/17.03 e later)

Antes de começar

Se você utilizou Trident em versões anteriores ao Docker 1.13/17.03 com o método tradicional de daemon, certifique-se de interromper o processo do Trident e reiniciar o daemon do Docker antes de usar o método de plugin gerenciado.

1. Interrompa todas as instâncias em execução:

```
pkill /usr/local/bin/netappdvp
pkill /usr/local/bin/trident
```

2. Reinicie Docker.

```
systemctl restart docker
```

3. Certifique-se de que você tenha o Docker Engine 17.03 (new 1.13) ou posterior instalado.

```
docker --version
```

Se a sua versão estiver desatualizada "instale ou atualize sua instalação".

Passos

1. Crie um arquivo de configuração e especifique as opções da seguinte forma:
   - `config`: O nome de arquivo padrão é `config.json`, porém você pode usar qualquer nome que escolher especificando a opção `config` com o nome do arquivo. O arquivo de configuração deve estar localizado no diretório `/etc/netappdvp` no sistema host.
   - `log-level`: Especifique o nível de registro (`debug`, `info`, `warn`, `error`, `fatal`). O padrão é `info`.
   - `debug`: especifique se o registro de depuração está ativado. O padrão é false. Substitui log-level se

true.

i. Crie um local para o arquivo de configuração:

```
sudo mkdir -p /etc/netappdvp
```

ii. Crie o arquivo de configuração:

```
cat << EOF > /etc/netappdvp/config.json
```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Inicie Trident usando o sistema de plugins gerenciados. Substitua `<version>` pela versão do plugin (xxx.xx.x) que você está usando.

```
docker plugin install --grant-all-permissions --alias netapp
netapp/trident-plugin:<version> config=myConfigFile.json
```

3. Comece a usar Trident para consumir storage do sistema configurado.

a. Crie um volume chamado "firstVolume":

```
docker volume create -d netapp --name firstVolume
```

b. Crie um volume padrão quando o contêiner for iniciado:

```
docker run --rm -it --volume-driver netapp --volume
secondVolume:/my_vol alpine ash
```

c. Remova o volume "firstVolume":

```
docker volume rm firstVolume
```

Método tradicional (versão 1.12 ou anterior)

Antes de começar

1. Certifique-se de ter Docker versão 1.10 ou posterior.

   ```
   docker --version
   ```

   Se a sua versão estiver desatualizada, atualize sua instalação.

   ```
   curl -fsSL https://get.docker.com/ | sh
   ```

   Ou, "Siga as instruções para a sua distribuição".

2. Certifique-se de que NFS e/ou iSCSI estejam configurados para o seu sistema.

Passos

1. Instale e configure o NetApp Docker Volume Plugin:
   a. Baixe e descompacte o aplicativo:

   ```
   wget
   https://github.com/NetApp/trident/releases/download/10.0/trident-
   installer-25.10.0.tar.gz
   tar zxf trident-installer-25.10.0.tar.gz
   ```

   b. Mova para um local no caminho do bin:

   ```
   sudo mv trident-installer/extras/bin/trident /usr/local/bin/
   sudo chown root:root /usr/local/bin/trident
   sudo chmod 755 /usr/local/bin/trident
   ```

   c. Crie um local para o arquivo de configuração:

   ```
   sudo mkdir -p /etc/netappdvp
   ```

   d. Crie o arquivo de configuração:

   ```
   cat << EOF > /etc/netappdvp/ontap-nas.json
   ```

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
EOF
```

2. Após colocar o arquivo binário e criar o arquivo de configuração, inicie o daemon Trident usando o arquivo de configuração desejado.

```
sudo trident --config=/etc/netappdvp/ontap-nas.json
```

> A menos que especificado, o nome padrão para o driver de volume é "netapp".

Após o daemon ser iniciado, você pode criar e gerenciar volumes usando a interface Docker CLI.

3. Crie um volume:

```
docker volume create -d netapp --name trident_1
```

4. Provisione um volume Docker ao iniciar um contêiner:

```
docker run --rm -it --volume-driver netapp --volume trident_2:/my_vol
alpine ash
```

5. Remova um volume do Docker:

```
docker volume rm trident_1
```

```
docker volume rm trident_2
```

Iniciar Trident na inicialização do sistema

Um arquivo de unidade de exemplo para sistemas baseados em systemd pode ser encontrado em `contrib/trident.service.example` no repositório Git. Para usar o arquivo com RHEL, faça o seguinte:

1. Copie o arquivo para o local correto.

   Você deve usar nomes exclusivos para os arquivos de unidade se tiver mais de uma instância em execução.

   ```
   cp contrib/trident.service.example
   /usr/lib/systemd/system/trident.service
   ```

2. Edite o arquivo, altere a descrição (linha 2) para corresponder ao nome do driver e o caminho do arquivo de configuração (linha 9) para refletir seu ambiente.
3. Recarregue o systemd para que ele incorpore as alterações:

   ```
   systemctl daemon-reload
   ```

4. Ative o serviço.

   Esse nome varia dependendo do nome que você deu ao arquivo no `/usr/lib/systemd/system` diretório.

   ```
   systemctl enable trident
   ```

5. Inicie o serviço.

   ```
   systemctl start trident
   ```

6. Veja o status.

   ```
   systemctl status trident
   ```

Sempre que você modificar o arquivo de unidade, execute o comando `systemctl daemon-reload` para que ele reconheça as alterações.

Atualize ou desinstale Trident

Você pode atualizar Trident para Docker com segurança, sem qualquer impacto nos volumes que estão em uso. Durante o processo de atualização, haverá um breve período em que `docker volume`comandos direcionados ao plugin não terão sucesso, e os aplicativos não poderão montar volumes até que o plugin esteja em execução novamente. Na maioria das circunstâncias, isso é uma questão de segundos.

Atualizar

Execute as etapas abaixo para atualizar o Trident para Docker.

Passos

1. Liste os volumes existentes:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

2. Desative o plugin:

```
docker plugin disable -f netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Atualize o plugin:

```
docker plugin upgrade --skip-remote-check --grant-all-permissions
netapp:latest netapp/trident-plugin:21.07
```

> A versão 18.01 do Trident substitui o nDVP. Você deve atualizar diretamente da `netapp/ndvp-plugin` imagem para a `netapp/trident-plugin` imagem.

4. Ative o plugin:

```
docker plugin enable netapp:latest
```

5. Verifique se o plugin está ativado:

```
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       Trident - NetApp Docker Volume
Plugin   true
```

6. Verifique se os volumes estão visíveis:

```
docker volume ls
DRIVER              VOLUME NAME
netapp:latest       my_volume
```

Se você estiver atualizando de uma versão antiga do Trident (anterior à 20.10) para o Trident 20.10 ou posterior, pode encontrar um erro. Para mais informações, consulte "Problemas conhecidos". Se você encontrar o erro, primeiro desative o plugin, depois remova o plugin e, em seguida, instale a versão necessária do Trident passando um parâmetro de configuração extra: `docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant-all-permissions config=config.json`

Desinstalar

Execute as etapas abaixo para desinstalar Trident para Docker.

Passos

1. Remova quaisquer volumes que o plugin criou.
2. Desative o plugin:

```
docker plugin disable netapp:latest
docker plugin ls
ID                  NAME                DESCRIPTION
ENABLED
7067f39a5df5        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
```

3. Remova o plugin:

```
docker plugin rm netapp:latest
```

Trabalhar com volumes

Você pode criar, clonar e remover volumes facilmente usando os comandos padrão `docker volume` com o nome do driver Trident especificado quando necessário.

Criar um volume

- Crie um volume com um driver usando o nome padrão:

```
docker volume create -d netapp --name firstVolume
```

- Crie um volume com uma instância específica do Trident:

```
docker volume create -d ntap_bronze --name bronzeVolume
```

Se você não especificar nenhum "opções", os padrões do driver serão usados.

- Substitua o tamanho de volume padrão. Veja o exemplo a seguir para criar um volume de 20 GiB com um driver:

```
docker volume create -d netapp --name my_vol --opt size=20G
```

Os tamanhos de volume são expressos como strings contendo um valor inteiro com unidades opcionais (exemplo: 10G, 20GB, 3TiB). Se nenhuma unidade for especificada, o padrão é G. As unidades de tamanho podem ser expressas como potências de 2 (B, KiB, MiB, GiB, TiB) ou potências de 10 (B, KB, MB, GB, TB). Unidades abreviadas usam potências de 2 (G = GiB, T = TiB, …).

Remover um volume

- Remova o volume da mesma forma que qualquer outro volume do Docker:

```
docker volume rm firstVolume
```

Ao usar o `solidfire-san` driver, o exemplo acima exclui e limpa o volume.

Execute as etapas abaixo para atualizar o Trident para Docker.

Clonar um volume

Ao usar os `ontap-nas`, `ontap-san` e `solidfire-san` drivers de storage, Trident pode clonar volumes. Ao usar os `ontap-nas-flexgroup` ou `ontap-nas-economy` drivers, a clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo snapshot.

- Inspecione o volume para enumerar snapshots:

```
docker volume inspect <volume_name>
```

- Crie um novo volume a partir de um volume existente. Isso resultará na criação de um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume>
```

- Crie um novo volume a partir de um snapshot existente em um volume. Isso não criará um novo snapshot:

```
docker volume create -d <driver_name> --name <new_name> -o from
=<source_docker_volume> -o fromSnapshot=<source_snap_name>
```

Exemplo

```
docker volume inspect firstVolume

[
  {
    "Driver": "ontap-nas",
    "Labels": null,
    "Mountpoint": "/var/lib/docker-volumes/ontap-
nas/netappdvp_firstVolume",
    "Name": "firstVolume",
    "Options": {},
    "Scope": "global",
    "Status": {
      "Snapshots": [
        {
          "Created": "2017-02-10T19:05:00Z",
          "Name": "hourly.2017-02-10_1505"
        }
      ]
    }
  }
]

docker volume create -d ontap-nas --name clonedVolume -o from=firstVolume
clonedVolume

docker volume rm clonedVolume
docker volume create -d ontap-nas --name volFromSnap -o from=firstVolume
-o fromSnapshot=hourly.2017-02-10_1505
volFromSnap

docker volume rm volFromSnap
```

Acessar volumes criados externamente

Você pode acessar dispositivos de bloco criados externamente (ou seus clones) por contêineres usando Trident **somente** se eles não tiverem partições e se o sistema de arquivos for compatível com Trident (por exemplo: um `ext4` formatado `/dev/sdc1` não será acessível via Trident).

Opções de volume específicas do driver

Cada driver de armazenamento possui um conjunto diferente de opções, que você pode especificar no momento da criação do volume para personalizar o resultado. Veja abaixo as opções que se aplicam ao seu sistema de storage configurado.

Utilizar essas opções durante a operação de criação de volume é simples. Forneça a opção e o valor usando o `-o` operador durante a operação da CLI. Esses valores substituem quaisquer valores equivalentes do arquivo de configuração JSON.

Opções de volume ONTAP

As opções de criação de volume para NFS, iSCSI e FC incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, padrão é 1 GiB.
`spaceReserve`	O provisionamento do volume pode ser fino ou espesso, sendo o padrão fino. Os valores válidos são `none` (provisionamento fino) e `volume` (provisionamento espesso).
`snapshotPolicy`	Isso definirá a política de snapshots para o valor desejado. O padrão é `none`, o que significa que nenhum snapshot será criado automaticamente para o volume. A menos que seja modificada pelo administrador de storage, existe uma política chamada "default" em todos os sistemas ONTAP que cria e retém seis snapshots por hora, dois por dia e dois por semana. Os dados preservados em um snapshot podem ser recuperados navegando até o diretório `.snapshot` em qualquer diretório do volume.
`snapshotReserve`	Isso definirá a reserva de snapshots para a porcentagem desejada. O padrão é nenhum valor, o que significa que ONTAP selecionará o snapshotReserve (geralmente 5%) se você tiver selecionado um snapshotPolicy, ou 0% se o snapshotPolicy for none. Você pode definir o valor padrão de snapshotReserve no arquivo de configuração para todos os backends ONTAP, e pode usá-lo como uma opção de criação de volume para todos os backends ONTAP, exceto o ontap-nas-economy.

Opção	Descrição
`splitOnClone`	Ao clonar um volume, isso fará com que ONTAP separe imediatamente o clone do seu volume pai. O padrão é `false`. Alguns casos de uso para clonagem de volumes são melhor atendidos separando o clone do seu volume pai imediatamente após a criação, pois é improvável que haja oportunidade para eficiências de armazenamento. Por exemplo, clonar um banco de dados vazio pode oferecer grande economia de tempo, mas pouca economia de armazenamento, então é melhor separar o clone imediatamente.
`encryption`	Ative NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e ativada no cluster para usar esta opção. Se o NAE estiver habilitado no backend, qualquer volume provisionado no Trident será habilitado para NAE. Para obter mais informações, consulte: "Como Trident funciona com NVE e NAE".
`tieringPolicy`	Define a política de hierarquização a ser usada para o volume. Isso decide se os dados serão movidos para a camada de nuvem quando se tornarem inativos (frios).

As seguintes opções adicionais são **exclusivas** para NFS:

Opção	Descrição
`unixPermissions`	Isso controla o conjunto de permissões para o próprio volume. Por padrão, as permissões serão definidas como `---rwxr-xr-x, ou em notação numérica 0755, e `root` será o proprietário. Tanto o formato de texto quanto o numérico funcionarão.
`snapshotDir`	Definir isso como `true` tornará o diretório `.snapshot` visível para clientes que acessam o volume. O valor padrão é `false`, o que significa que a visibilidade do diretório `.snapshot` está desativada por padrão. Algumas imagens, por exemplo, a imagem oficial do MySQL, não funcionam como esperado quando o diretório `.snapshot` está visível.
`exportPolicy`	Define a política de exportação a ser usada para o volume. O padrão é `default`.

Opção	Descrição
`securityStyle`	Define o estilo de segurança a ser usado para acesso ao volume. O padrão é `unix`. Os valores válidos são `unix` e `mixed`.

As seguintes opções adicionais são **apenas** para iSCSI:

Opção	Descrição
`fileSystemType`	Define o sistema de arquivos usado para formatar volumes iSCSI. O padrão é `ext4`. Os valores válidos são `ext3`, `ext4` e `xfs`.
`spaceAllocation`	Definir isto para `false` desativará o recurso de alocação de espaço do LUN. O valor padrão é `true`, o que significa que ONTAP notifica o host quando o volume fica sem espaço e o LUN no volume não pode aceitar gravações. Esta opção também permite que ONTAP recupere espaço automaticamente quando o host exclui dados.

Exemplos

Veja os exemplos abaixo:

- Crie um volume de 10 GiB:

```
docker volume create -d netapp --name demo -o size=10G -o
encryption=true
```

- Crie um volume de 100 GiB com snapshots:

```
docker volume create -d netapp --name demo -o size=100G -o
snapshotPolicy=default -o snapshotReserve=10
```

- Crie um volume com o bit setUID ativado:

```
docker volume create -d netapp --name demo -o unixPermissions=4755
```

O tamanho mínimo do volume é 20 MiB.

Se a reserva de snapshots não for especificada e a política de snapshots for `none`, Trident usa uma reserva de snapshots de 0%.

- Crie um volume sem política de snapshot e sem reserva de snapshot:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
```

- Crie um volume sem política de snapshots e com uma reserva de snapshots personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt snapshotPolicy=none
--opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e uma reserva de snapshot personalizada de 10%:

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy --opt snapshotReserve=10
```

- Crie um volume com uma política de snapshot e aceite a reserva de snapshot padrão do ONTAP (normalmente 5%):

```
docker volume create -d netapp --name my_vol --opt
snapshotPolicy=myPolicy
```

Opções de volume do software Element

As opções do software Element expõem o tamanho e as políticas de qualidade do serviço (QoS) associadas ao volume. Quando o volume é criado, a política de QoS associada a ele é especificada usando a `-o type=service_level` nomenclatura.

O primeiro passo para definir um nível de serviço de qualidade do serviço com o driver Element é criar pelo menos um tipo e especificar os valores mínimo, máximo e de burst de IOPS associados a um nome no arquivo de configuração.

Outras opções de criação de volume do software Element incluem o seguinte:

Opção	Descrição
`size`	O tamanho do volume, o padrão é 1 GiB ou entrada de configuração … "defaults": {"size": "5G"}.
`blocksize`	Use 512 ou 4096, padrão é 512 ou entrada de configuração DefaultBlockSize.

Exemplo

Veja o seguinte arquivo de configuração com definições de QoS:

```
{
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

Na configuração acima, temos três definições de política: Bronze, Silver e Gold. Esses nomes são arbitrários.

- Crie um volume Gold de 10 GiB:

```
docker volume create -d solidfire --name sfGold -o type=Gold -o size=10G
```

- Crie um volume Bronze de 100 GiB:

```
docker volume create -d solidfire --name sfBronze -o type=Bronze -o
size=100G
```

Coletar registros

Você pode coletar registros para ajudar na solução de problemas. O método que você

usa para coletar os registros varia de acordo com a forma como você está executando o plugin do Docker.

Colete registros para solucionar problemas

Passos

1. Se você estiver executando Trident usando o método de plugin gerenciado recomendado (ou seja, usando `docker plugin` comandos), visualize-os da seguinte forma:

```
docker plugin ls
```

```
ID                  NAME                DESCRIPTION
ENABLED
4fb97d2b956b        netapp:latest       nDVP - NetApp Docker Volume
Plugin   false
journalctl -u docker | grep 4fb97d2b956b
```

   O nível de registro padrão deve permitir que você diagnostique a maioria dos problemas. Se achar que isso não é suficiente, você pode ativar o registro de depuração.

2. Para ativar o registro de depuração, instale o plugin com o registro de depuração ativado:

```
docker plugin install netapp/trident-plugin:<version> --alias <alias>
debug=true
```

   Ou ative o registro de depuração quando o plugin já estiver instalado:

```
docker plugin disable <plugin>
```

```
docker plugin set <plugin> debug=true
```

```
docker plugin enable <plugin>
```

3. Se você estiver executando o próprio binário no host, os logs estarão disponíveis no diretório `/var/log/netappdvp` do host. Para ativar o registro de depuração, especifique `-debug` ao executar o plugin.

Dicas gerais de resolução de problemas

- O problema mais comum que os novos usuários enfrentam é uma configuração incorreta que impede o plugin de inicializar. Quando isso acontece, você provavelmente verá uma mensagem como esta ao tentar instalar ou habilitar o plugin:

`Error response from daemon: dial unix /run/docker/plugins/<id>/netapp.sock: connect: no such file or directory`

Isso significa que o plugin falhou ao iniciar. Felizmente, o plugin foi desenvolvido com um recurso de registro abrangente que deve ajudá-lo a diagnosticar a maioria dos problemas que você provavelmente encontrará.

- Se houver problemas ao montar um PV em um contêiner, verifique se `rpcbind` está instalado e em execução. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status do serviço rpcbind executando um `systemctl status rpcbind` ou equivalente.

Gerenciar várias instâncias do Trident

São necessárias várias instâncias do Trident quando se deseja ter várias configurações de storage disponíveis simultaneamente. A chave para ter várias instâncias é atribuir nomes diferentes a elas usando a opção `--alias` com o plugin containerizado, ou a opção `--volume-driver` ao instanciar Trident no host.

Etapas para o plugin gerenciado do Docker (versão 1.13/17.03 ou later)

1. Inicie a primeira instância especificando um alias e arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias silver
netapp/trident-plugin:21.07 config=silver.json
```

2. Inicie a segunda instância, especificando um alias diferente e um arquivo de configuração.

```
docker plugin install --grant-all-permissions --alias gold
netapp/trident-plugin:21.07 config=gold.json
```

3. Crie volumes especificando o alias como o nome do driver.

   Por exemplo, para o volume gold:

```
docker volume create -d gold --name ntapGold
```

   Por exemplo, para o volume silver:

```
docker volume create -d silver --name ntapSilver
```

Passos para tradicional (versão 1.12 ou anterior)

1. Inicie o plugin com uma configuração NFS usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-nas --config=/path/to/config
-nfs.json
```

2. Inicie o plugin com uma configuração iSCSI usando um ID de driver personalizado:

```
sudo trident --volume-driver=netapp-san --config=/path/to/config
-iscsi.json
```

3. Provisione volumes Docker para cada instância de driver:

   Por exemplo, para NFS:

```
docker volume create -d netapp-nas --name my_nfs_vol
```

   Por exemplo, para iSCSI:

```
docker volume create -d netapp-san --name my_iscsi_vol
```

Opções de configuração de storage

Veja as opções de configuração disponíveis para suas configurações do Trident.

Opções globais de configuração

Essas opções de configuração se aplicam a todas as configurações do Trident, independentemente da plataforma de storage utilizada.

Opção	Descrição	Exemplo
`version`	Número da versão do arquivo de configuração	`1`
`storageDriverName`	Nome do driver de armazenamento	`ontap-nas`, `ontap-san`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `solidfire-san`
`storagePrefix`	Prefixo opcional para nomes de volumes. Padrão: `netappdvp_`.	`staging_`
`limitVolumeSize`	Restrição opcional nos tamanhos de volume. Padrão: "" (não aplicada)	`10g`

Não use `storagePrefix` (incluindo o padrão) para backends do Element. Por padrão, o driver `solidfire-san` ignorará essa configuração e não usará um prefixo. NetApp recomenda usar um tenantID específico para o mapeamento de volumes do Docker ou usar o atributo data, que é preenchido com a versão do Docker, informações do driver e o nome bruto do Docker nos casos em que qualquer manipulação de nomes possa ter sido usada.

As opções padrão estão disponíveis para evitar a necessidade de especificá-las em cada volume que você criar. A opção `size` está disponível para todos os tipos de controladores. Consulte a seção de configuração do ONTAP para um exemplo de como definir o tamanho padrão do volume.

Opção	Descrição	Exemplo
`size`	Tamanho padrão opcional para novos volumes. Padrão: `1G`	`10G`

Configuração do ONTAP

Além dos valores de configuração global acima, ao usar ONTAP, as seguintes opções de nível superior estão disponíveis.

Opção	Descrição	Exemplo
`managementLIF`	Endereço IP da LIF de gerenciamento do ONTAP. Você pode especificar um nome de domínio totalmente qualificado (FQDN).	`10.0.0.1`
`dataLIF`	Endereço IP do protocolo LIF. **ONTAP NAS drivers**: NetApp recomenda especificar `dataLIF`. Se não for fornecido, Trident busca os dataLIFs da SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. **ONTAP SAN drivers**: Não especifique para iSCSI ou FC. Trident usa "Mapa de LUN seletivo do ONTAP" para descobrir as LIFs iSCSI ou FC necessárias para estabelecer uma sessão de múltiplos caminhos. Um aviso é gerado se `dataLIF` for explicitamente definido.	`10.0.0.2`

Opção	Descrição	Exemplo
`svm`	Storage virtual machine a ser usada (obrigatória se a LIF de gerenciamento for uma LIF de cluster)	`svm_nfs`
`username`	Nome de usuário para conectar ao dispositivo de storage	`vsadmin`
`password`	Senha para conectar ao dispositivo de storage	`secret`
`aggregate`	Agregado para provisionamento (opcional; se definido, deve ser atribuído à SVM). Para o `ontap-nas-flexgroup` driver, esta opção é ignorada. Todos os agregados atribuídos à SVM são usados para provisionar um FlexGroup volume.	`aggr1`
`limitAggregateUsage`	Opcional, falhe o provisionamento se a utilização estiver acima desta percentagem	`75%`
`nfsMountOptions`	Controle preciso das opções de montagem NFS; o padrão é "-o nfsvers=3". **Disponível apenas para os `ontap-nas` e `ontap-nas-economy` drivers**. "Veja as informações de configuração do host NFS aqui".	`-o nfsvers=4`
`igroupName`	Trident cria e gerencia por nó `igroups` como `netappdvp`. Este valor não pode ser alterado nem omitido. **Disponível apenas para o `ontap-san` driver**.	`netappdvp`
`limitVolumeSize`	Tamanho máximo de volume solicitável.	`300g`

Opção	Descrição	Exemplo
`qtreesPerFlexvol`	Máximo de qtrees por FlexVol, deve estar no intervalo [50, 300], padrão é 200. **Para o `ontap-nas-economy` driver, esta opção permite personalizar o número máximo de qtrees por FlexVol**.	`300`
`sanType`	**Compatível para `ontap-san` driver apenas.** Use para selecionar `iscsi` para iSCSI, `nvme` para NVMe/TCP ou `fcp` para SCSI sobre Fibre Channel (FC).	`iscsi` se estiver em branco
`limitVolumePoolSize`	**Compatível para `ontap-san-economy` e `ontap-san-economy` drivers apenas.** Limita os tamanhos do FlexVol nos drivers ONTAP ontap-nas-economy e ontap-SAN-economy.	`300g`

As opções padrão estão disponíveis para evitar que você precise especificá-las em cada volume que você criar:

Opção	Descrição	Exemplo
`spaceReserve`	Modo de reserva de espaço; `none` (provisionamento dinâmico) ou `volume` (provisionamento robusto)	`none`
`snapshotPolicy`	Política do Snapshot a ser usada, o padrão é `none`	`none`
`snapshotReserve`	Percentual de reserva do Snapshot, o padrão é "" para aceitar o padrão do ONTAP	`10`
`splitOnClone`	Separar um clone de seu pai no momento da criação, o padrão é `false`	`false`
`encryption`	Habilita a NetApp Volume Encryption (NVE) no novo volume; o padrão é `false`. A NVE deve estar licenciada e habilitada no cluster para usar esta opção. Se o NAE estiver habilitado no backend, qualquer volume provisionado no Trident será habilitado para NAE. Para obter mais informações, consulte: "Como Trident funciona com NVE e NAE".	verdadeiro

Opção	Descrição	Exemplo
`unixPermissions`	Opção NAS para volumes NFS provisionados, o padrão é `777`	`777`
`snapshotDir`	Opção NAS para acesso ao `.snapshot` diretório.	"true" para NFSv4 "false" para NFSv3
`exportPolicy`	Opção NAS para a política de exportação NFS a ser usada, padrão é `default`	`default`
`securityStyle`	Opção NAS para acesso ao volume NFS provisionado. NFS suporta `mixed` e `unix` estilos de segurança. O padrão é `unix`.	`unix`
`fileSystemType`	Opção SAN para selecionar o tipo de sistema de arquivos, o padrão é `ext4`	`xfs`
`tieringPolicy`	Política de tiering a ser usada, padrão é `none`.	`none`
`skipRecoveryQueue`	Durante a exclusão de volume, ignore a fila de recuperação no armazenamento e exclua o volume imediatamente.	``

Opções de dimensionamento

Os `ontap-nas` e `ontap-san` drivers criam um ONTAP FlexVol para cada volume Docker. ONTAP suporta até 1000 FlexVols por nó de cluster, com um máximo de 12.000 volumes FlexVol por cluster. Se os requisitos de volume Docker se enquadrarem nessa limitação, o driver `ontap-nas` é a solução NAS preferida devido aos recursos adicionais oferecidos por FlexVols, como snapshots granulares de volume Docker e clonagem.

Se você precisar de mais volumes Docker do que os limites do FlexVol permitem, escolha o `ontap-nas-economy` ou o `ontap-san-economy` driver.

O `ontap-nas-economy` driver cria volumes Docker como Qtrees ONTAP dentro de um pool de FlexVol volumes gerenciados automaticamente. Qtrees oferecem escalabilidade muito maior, até 100.000 por nó de cluster e 2.400.000 por cluster, à custa de alguns recursos. O `ontap-nas-economy` driver não suporta snapshots granulares de volume Docker nem clonagem.

> O `ontap-nas-economy`driver não é atualmente suportado em Docker Swarm, porque Docker Swarm não orquestra a criação de volumes em vários nós.

O `ontap-san-economy` driver cria volumes Docker como LUNs ONTAP dentro de um pool compartilhado de FlexVol volumes gerenciados automaticamente. Dessa forma, cada FlexVol não fica restrito a apenas um LUN e oferece melhor escalabilidade para cargas de trabalho SAN. Dependendo do array de storage, ONTAP suporta até 16384 LUNs por cluster. Como os volumes são LUNs, este driver suporta snapshots e clonagem granulares de volumes Docker.

Escolha o `ontap-nas-flexgroup` driver para aumentar o paralelismo em um único volume que pode crescer até a faixa de petabytes com bilhões de arquivos. Alguns casos de uso ideais para FlexGroups incluem IA/ML/DL, big data e analytics, builds de software, streaming, repositórios de arquivos e assim por diante. Trident usa todos os agregados atribuídos a uma SVM ao provisionar um volume FlexGroup. O suporte a FlexGroup no Trident também tem as seguintes considerações:

- Requer versão do ONTAP 9.2 ou superior.
- Até o momento, FlexGroups só suportam NFS v3.
- Recomenda-se habilitar os identificadores NFSv3 de 64 bits para a SVM.
- O tamanho mínimo recomendado para membros/volumes FlexGroup é de 100 GiB.
- A clonagem não é suportada para volumes FlexGroup.

Para obter informações sobre FlexGroups e cargas de trabalho que são apropriadas para FlexGroups, consulte o "NetApp FlexGroup volume Guia de Melhores Práticas e Implementação".

Para obter recursos avançados e grande escala no mesmo ambiente, você pode executar várias instâncias do Docker Volume Plugin, uma usando `ontap-nas` e outra usando `ontap-nas-economy`.

Função ONTAP personalizada para Trident

Você pode criar uma função de cluster ONTAP com privilégios mínimos para que não precise usar a função de administrador do ONTAP para executar operações no Trident. Ao incluir o nome de usuário em um arquivo de configuração de backend do Trident, o Trident usa a função de cluster ONTAP que você criou para executar as operações.

Consulte "Gerador de funções personalizadas Trident" para obter mais informações sobre como criar funções personalizadas do Trident.

Usando ONTAP CLI

1. Crie uma nova função usando o seguinte comando:

```
security login role create <role_name\> -cmddirname "command" -access all
-vserver <svm_name\>
```

2. Crie um nome de usuário para o usuário do Trident:

```
security login create -username <user_name\> -application ontapi
-authmethod password -role <name_of_role_in_step_1\> -vserver <svm_name\>
-comment "user_description"
security login create -username <user_name\> -application http -authmethod
password -role <name_of_role_in_step_1\> -vserver <svm_name\> -comment
"user_description"
```

3. Mapeie a função para o usuário:

```
security login modify username <user_name\> -vserver <svm_name\> -role
<role_name\> -application ontapi -application console -authmethod
<password\>
```

Usando System Manager

Execute as seguintes etapas no ONTAP System Manager:

1. **Crie uma função personalizada**:
   a. Para criar uma função personalizada no nível do cluster, selecione **Cluster > Settings**.

      (Ou) Para criar uma função personalizada no nível da SVM, selecione **Storage > Storage VMs > `required SVM`> Settings > Users and Roles**.

   b. Selecione o ícone de seta (→) ao lado de **Users and Roles**.
   c. Selecione **+Adicionar** em **Roles**.
   d. Defina as regras para a função e clique em **Save**.
2. **Mapeie a função ao usuário Trident**: + Execute as seguintes etapas na página **Usuários e Funções**:
   a. Selecione o ícone Adicionar **+** em **Usuários**.
   b. Selecione o nome de usuário desejado e selecione uma função no menu suspenso para **Função**.
   c. Clique em **Salvar**.

Consulte as seguintes páginas para obter mais informações:

- "Funções personalizadas para administração do ONTAP" ou "Definir funções personalizadas"
- "Trabalhe com funções e usuários"

Exemplos de arquivos de configuração do ONTAP

Exemplo de NFS para <code>ontap-nas</code> driver

```
{
  "version": 1,
  "storageDriverName": "ontap-nas",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "defaults": {
    "size": "10G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

Exemplo de NFS para <code>ontap-nas-flexgroup</code> driver

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-flexgroup",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "defaults": {
    "size": "100G",
    "spaceReserve": "none",
    "exportPolicy": "default"
  }
}
```

Exemplo de NFS para o driver <code>ontap-nas-economy</code>

```
{
  "version": 1,
  "storageDriverName": "ontap-nas-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.2",
  "svm": "svm_nfs",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1"
}
```

Exemplo de iSCSI para <code>ontap-san</code> driver

```
{
  "version": 1,
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

Exemplo de NFS para <code>ontap-san-economy</code> driver

```
{
  "version": 1,
  "storageDriverName": "ontap-san-economy",
  "managementLIF": "10.0.0.1",
  "dataLIF": "10.0.0.3",
  "svm": "svm_iscsi_eco",
  "username": "vsadmin",
  "password": "password",
  "aggregate": "aggr1",
  "igroupName": "netappdvp"
}
```

Exemplo de NVMe/TCP para <code>ontap-san</code> driver

```
{
  "version": 1,
  "backendName": "NVMeBackend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "svm": "svm_nvme",
  "username": "vsadmin",
  "password": "password",
  "sanType": "nvme",
  "useREST": true
}
```

Exemplo de SCSI sobre FC para <code>ontap-san</code> driver

```
{
  "version": 1,
  "backendName": "ontap-san-backend",
  "storageDriverName": "ontap-san",
  "managementLIF": "10.0.0.1",
  "sanType": "fcp",
  "svm": "trident_svm",
  "username": "vsadmin",
  "password": "password",
  "useREST": true
}
```

Configuração do software Element

Além dos valores de configuração global, ao usar o software Element (NetApp HCI/SolidFire), essas opções estão disponíveis.

Opção	Descrição	Exemplo
`Endpoint`	https://<login>:<password>@<mvip>/json-rpc/<element-version>	https://admin:admin@192.168.160.3/json-rpc/8.0
`SVIP`	endereço IP iSCSI e porta	10.0.0.7:3260
`TenantName`	SolidFireF Tenant a ser usado (criado se não for encontrado)	`docker`

Opção	Descrição	Exemplo
`InitiatorIFace`	Especifique a interface ao restringir o tráfego iSCSI para uma interface não padrão	`default`
`Types`	Especificações de QoS	Veja o exemplo abaixo
`LegacyNamePrefix`	Prefixo para instalações atualizadas do Trident. Se você usou uma versão do Trident anterior à 1.3.2 e realizar uma atualização com volumes existentes, será necessário definir esse valor para acessar seus volumes antigos que foram mapeados pelo método de nome do volume.	`netappdvp-`

O `solidfire-san` driver não é compatível com Docker Swarm.

Exemplo de arquivo de configuração do Element software

```
{
  "version": 1,
  "storageDriverName": "solidfire-san",
  "Endpoint": "https://admin:admin@192.168.160.3/json-rpc/8.0",
  "SVIP": "10.0.0.7:3260",
  "TenantName": "docker",
  "InitiatorIFace": "default",
  "Types": [
    {
      "Type": "Bronze",
      "Qos": {
        "minIOPS": 1000,
        "maxIOPS": 2000,
        "burstIOPS": 4000
      }
    },
    {
      "Type": "Silver",
      "Qos": {
        "minIOPS": 4000,
        "maxIOPS": 6000,
        "burstIOPS": 8000
      }
    },
    {
      "Type": "Gold",
      "Qos": {
        "minIOPS": 6000,
        "maxIOPS": 8000,
        "burstIOPS": 10000
      }
    }
  ]
}
```

Problemas e limitações conhecidos

Encontre informações sobre problemas e limitações conhecidos ao usar Trident com Docker.

A atualização do Trident Docker Volume Plugin para a versão 20.10 ou posterior a partir de versões mais antigas resulta em falha na atualização com o erro "arquivo ou diretório inexistente".

Solução alternativa

1. Desativar o plugin.

```
docker plugin disable -f netapp:latest
```

2. Remova o plugin.

```
docker plugin rm -f netapp:latest
```

3. Reinstale o plugin fornecendo o parâmetro adicional `config`.

```
docker plugin install netapp/trident-plugin:20.10 --alias netapp --grant
-all-permissions config=config.json
```

Os nomes dos volumes devem ter no mínimo 2 caracteres de comprimento.

Esta é uma limitação do cliente Docker. O cliente interpretará um nome de caractere único como sendo um caminho do Windows "Veja o bug 25773".

Docker Swarm possui certos comportamentos que impedem Trident de oferecer suporte a todas as combinações de storage e driver.

- Atualmente, Docker Swarm utiliza o nome do volume em vez do ID do volume como seu identificador de volume exclusivo.
- As solicitações de volume são enviadas simultaneamente para cada nó em um cluster Swarm.
- Os plugins de volume (incluindo Trident) devem ser executados independentemente em cada nó de um cluster Swarm. Devido à forma como o ONTAP funciona e como os `ontap-nas` e `ontap-san`drivers funcionam, eles são os únicos que conseguem operar dentro dessas limitações.

Os demais drivers estão sujeitos a problemas como condições de corrida, que podem resultar na criação de um grande número de volumes para uma única solicitação sem um "vencedor" claro; por exemplo, Element possui um recurso que permite que os volumes tenham o mesmo nome, mas IDs diferentes.

NetApp forneceu feedback à equipe do Docker, mas não tem nenhuma indicação de recurso futuro.

Se um FlexGroup estiver sendo provisionado, ONTAP não provisiona um segundo FlexGroup se o segundo FlexGroup tiver um ou mais agregados em comum com o FlexGroup que está sendo provisionado.

Melhores práticas e recomendações

Implantação

Use as recomendações listadas aqui ao implantar Trident.

Implantar em um namespace dedicado

"Espaços de nomes" fornecem separação administrativa entre diferentes aplicações e são uma barreira para o compartilhamento de recursos. Por exemplo, um PVC de um namespace não pode ser consumido por outro. Trident fornece recursos de PV para todos os namespaces no cluster Kubernetes e, consequentemente, utiliza uma conta de serviço que possui privilégios elevados.

Além disso, o acesso ao Trident pod pode permitir que um usuário acesse as credenciais do sistema de storage e outras informações confidenciais. É importante garantir que os usuários do aplicativo e os aplicativos de gerenciamento não tenham a capacidade de acessar as definições de objeto do Trident ou os próprios pods.

Utilize quotas e limites de intervalo para controlar o consumo de armazenamento

O Kubernetes possui dois recursos que, quando combinados, fornecem um mecanismo poderoso para limitar o consumo de recursos por aplicativos. O "mecanismo de cota de armazenamento" permite que o administrador implemente limites globais e específicos de classe de armazenamento para capacidade e contagem de objetos por namespace. Além disso, usar um "limite de intervalo" garante que as solicitações de PVC estejam dentro de um valor mínimo e máximo antes de serem encaminhadas ao provisionador.

Esses valores são definidos para cada namespace individualmente, o que significa que cada namespace deve ter valores definidos que estejam de acordo com seus requisitos de recursos. Veja aqui para informações sobre "como aproveitar quotas".

Configuração de storage

Cada plataforma de storage no portfólio da NetApp possui recursos exclusivos que beneficiam aplicações, conteinerizadas ou não.

Visão geral da plataforma

Trident funciona com ONTAP e Element. Não existe uma plataforma que seja mais adequada para todas as aplicações e cenários do que outra, no entanto, as necessidades da aplicação e da equipe que administra o dispositivo devem ser levadas em consideração ao escolher uma plataforma.

Você deve seguir as melhores práticas básicas do sistema operacional com o protocolo que está utilizando. Opcionalmente, você pode considerar incorporar as melhores práticas da aplicação, quando disponíveis, com configurações de backend, storage class e PVC para otimizar o storage para aplicações específicas.

ONTAP e Cloud Volumes ONTAP melhores práticas

Aprenda as melhores práticas para configurar ONTAP e Cloud Volumes ONTAP para Trident.

As recomendações a seguir são diretrizes para configurar ONTAP para cargas de trabalho conteinerizadas, que consomem volumes provisionados dinamicamente pelo Trident. Cada uma deve ser considerada e

avaliada quanto à sua adequação ao seu ambiente.

Use SVM(s) dedicada(s) ao Trident

As Storage Virtual Machines (SVMs) fornecem isolamento e separação administrativa entre os tenants em um sistema ONTAP. Dedicar uma SVM a aplicativos permite a delegação de privilégios e possibilita aplicar as melhores práticas para limitar o consumo de recursos.

Existem diversas opções disponíveis para o gerenciamento da SVM:

- Forneça a interface de gerenciamento do cluster na configuração do backend, juntamente com as credenciais apropriadas, e especifique o nome da SVM.
- Crie uma interface de gerenciamento dedicada para a SVM usando ONTAP System Manager ou a CLI.
- Compartilhe a função de gerenciamento com uma interface de dados NFS.

Em cada caso, a interface deve estar no DNS e o nome DNS deve ser usado ao configurar Trident. Isso ajuda a facilitar alguns cenários de DR, por exemplo, SVM-DR sem o uso de retenção de identidade de rede.

Não há preferência entre ter uma LIF de gerenciamento dedicada ou compartilhada para a SVM, no entanto, você deve garantir que suas políticas de segurança de rede estejam alinhadas com a abordagem que você escolher. Independentemente disso, a LIF de gerenciamento deve ser acessível via DNS para facilitar a máxima flexibilidade caso "SVM-DR" seja usada em conjunto com Trident.

Limite a contagem máxima de volume

Os sistemas de storage ONTAP têm uma contagem de volume máxima, que varia de acordo com a versão do software e a plataforma de hardware. Consulte "NetApp Hardware Universe" para sua plataforma específica e versão do ONTAP para determinar os limites exatos. Quando a contagem de volume é atingida, as operações de provisionamento falham não apenas para Trident, mas para todas as solicitações de storage.

Os drivers do Trident `ontap-nas` e `ontap-san` provisionam um FlexVolume para cada Persistent Volume (PV) do Kubernetes criado. O driver `ontap-nas-economy` cria aproximadamente um FlexVolume para cada 200 PVs (configurável entre 50 e 300). O driver `ontap-san-economy` cria aproximadamente um FlexVolume para cada 100 PVs (configurável entre 50 e 200). Para evitar que o Trident consuma todos os volumes disponíveis no sistema de storage, você deve definir um limite no SVM. Você pode fazer isso pela linha de comando:

```
vserver modify -vserver <svm_name> -max-volumes <num_of_volumes>
```

O valor para `max-volumes` varia de acordo com diversos critérios específicos do seu ambiente:

- O número de volumes existentes no cluster ONTAP
- O número de volumes que você espera provisionar fora do Trident para outros aplicativos
- O número de volumes persistentes que se espera que sejam consumidos por aplicações Kubernetes

O `max-volumes` valor é o total de volumes provisionados em todos os nós do cluster ONTAP, e não em um nó ONTAP individual. Como resultado, você pode encontrar algumas situações em que um nó de cluster ONTAP pode ter muito mais ou muito menos volumes provisionados pelo Trident do que outro nó.

Por exemplo, um cluster ONTAP de dois nós tem a capacidade de hospedar um máximo de 2000 FlexVol volumes. Ter a contagem máxima de volume definida para 1250 parece muito razoável. No entanto, se

apenas "agregados" de um nó forem atribuídos à SVM, ou se os agregados atribuídos de um nó não puderem ser provisionados (por exemplo, devido à capacidade), então o outro nó se torna o destino para todos os volumes provisionados pelo Trident. Isso significa que o limite de volume pode ser atingido para esse nó antes que o valor `max-volumes` seja alcançado, resultando em impacto tanto para o Trident quanto para outras operações de volume que utilizam esse nó. **Você pode evitar essa situação garantindo que os agregados de cada nó do cluster sejam atribuídos à SVM usada pelo Trident em números iguais.**

Clonar um volume

NetApp Trident suporta clonagem de volumes ao usar os `ontap-nas`, `ontap-san` e `solidfire-san` drivers de armazenamento. Ao usar os `ontap-nas-flexgroup` ou `ontap-nas-economy` drivers, a clonagem não é suportada. Criar um novo volume a partir de um volume existente resultará na criação de um novo snapshot.

Evite clonar um PVC associado a um StorageClass diferente. Realize operações de clonagem dentro do mesmo StorageClass para garantir a compatibilidade e evitar comportamentos inesperados.

Limite o tamanho máximo dos volumes criados pelo Trident

Para configurar o tamanho máximo para volumes que podem ser criados pelo Trident, use o `limitVolumeSize` parâmetro na sua `backend.json` definição.

Além de controlar o tamanho do volume no array de storage, você também deve aproveitar as capacidades do Kubernetes.

Limite o tamanho máximo de FlexVols criados pelo Trident

Para configurar o tamanho máximo dos FlexVols usados como pools para os drivers ontap-san-economy e ontap-nas-economy, use o `limitVolumePoolSize` parâmetro na sua `backend.json` definição.

Configurar o Trident para usar CHAP bidirecional

Você pode especificar os nomes de usuário e senhas do iniciador e do destino CHAP na definição do seu backend e fazer com que Trident habilite o CHAP na SVM. Usando o `useCHAP` parâmetro na configuração do seu backend, Trident autentica as conexões iSCSI para backends ONTAP com CHAP.

Crie e use uma política de QoS para SVM

Ao utilizar uma política de qualidade do serviço (QoS) do ONTAP, aplicada à SVM, limita-se o número de IOPS consumíveis pelos volumes provisionados pelo Trident. Isso ajuda a "prevenir um agressor" evitar que um contêiner descontrolado afete cargas de trabalho fora da SVM do Trident.

Você pode criar uma política de QoS para a SVM em algumas etapas. Consulte a documentação da sua versão do ONTAP para as informações mais precisas. O exemplo abaixo cria uma política de QoS que limita o total de IOPS disponível para a SVM a 5000.

```
# create the policy group for the SVM
qos policy-group create -policy-group <policy_name> -vserver <svm_name>
-max-throughput 5000iops

# assign the policy group to the SVM, note this will not work
# if volumes or files in the SVM have existing QoS policies
vserver modify -vserver <svm_name> -qos-policy-group <policy_name>
```

Além disso, se a sua versão do ONTAP for compatível, você pode considerar o uso de um QoS mínimo para garantir uma quantidade de largura de banda para cargas de trabalho em contêineres. O QoS adaptável não é compatível com uma política de nível SVM.

O número de IOPS dedicados às cargas de trabalho em contêineres depende de muitos aspectos. Entre outras coisas, incluem:

- Outras cargas de trabalho que utilizam o array de storage. Caso existam outras cargas de trabalho, não relacionadas à implantação do Kubernetes, que utilizem os recursos de storage, deve-se tomar cuidado para garantir que essas cargas de trabalho não sejam acidentalmente afetadas negativamente.
- Cargas de trabalho esperadas em execução em contêineres. Se cargas de trabalho com altos requisitos de IOPS forem executadas em contêineres, uma política de QoS baixa resulta em uma experiência ruim.

É importante lembrar que uma política de QoS atribuída no nível da SVM resulta em todos os volumes provisionados para a SVM compartilharem o mesmo pool de IOPS. Se uma, ou um pequeno número, das aplicações conteinerizadas tiver uma alta necessidade de IOPS, ela pode se tornar um problema para as outras cargas de trabalho conteinerizadas. Se for esse o caso, talvez você queira considerar o uso de automação externa para atribuir políticas de QoS por volume.

> Você deve atribuir o grupo de políticas de qualidade do serviço (QoS) à SVM **somente** se a sua versão do ONTAP for anterior à 9.8.

Criar grupos de políticas de QoS para Trident

A qualidade do serviço (QoS) garante que o desempenho de cargas de trabalho críticas não seja prejudicado por cargas de trabalho concorrentes. Os grupos de políticas de QoS do ONTAP fornecem opções de QoS para volumes e permitem que os usuários definam o limite de taxa de transferência para uma ou mais cargas de trabalho. Para mais informações sobre QoS, consulte "Garantindo a taxa de transferência com qualidade do serviço". Você pode especificar grupos de políticas de QoS no backend ou em um pool de storage, e eles são aplicados a cada volume criado nesse pool ou backend.

ONTAP possui dois tipos de grupos de políticas de qualidade do serviço: tradicional e adaptável. Os grupos de políticas tradicionais fornecem uma taxa máxima (ou mínima, em versões posteriores) de IOPS fixa. A qualidade do serviço adaptável dimensiona automaticamente a taxa de transferência de acordo com o tamanho da carga de trabalho, mantendo a proporção de IOPS para TBs|GBs conforme o tamanho da carga de trabalho muda. Isso proporciona uma vantagem significativa ao gerenciar centenas ou milhares de cargas de trabalho em uma grande implementação.

Considere o seguinte ao criar grupos de políticas de QoS:

- Você deve definir a `qosPolicy` chave no bloco `defaults` da configuração do backend. Veja o seguinte exemplo de configuração do backend:

```
---
version: 1
storageDriverName: ontap-nas
managementLIF: 0.0.0.0
dataLIF: 0.0.0.0
svm: svm0
username: user
password: pass
defaults:
  qosPolicy: standard-pg
storage:
  - labels:
      performance: extreme
    defaults:
      adaptiveQosPolicy: extremely-adaptive-pg
  - labels:
      performance: premium
    defaults:
      qosPolicy: premium-pg
```

- Você deve aplicar os grupos de políticas por volume, para que cada volume receba toda a taxa de transferência especificada pelo grupo de políticas. Grupos de políticas compartilhados não são suportados.

Para obter mais informações sobre grupos de políticas de qualidade do serviço (QoS), consulte "Referência de comandos ONTAP".

Limite o acesso aos recursos de storage aos membros do cluster Kubernetes

Limitar o acesso aos volumes NFS, LUNs iSCSI e LUNs FC criados pelo Trident é um componente crítico da postura de segurança da sua implementação do Kubernetes. Isso impede que hosts que não fazem parte do cluster Kubernetes acessem os volumes e potencialmente modifiquem dados inesperadamente.

É importante entender que os namespaces são o limite lógico para os recursos no Kubernetes. A premissa é que recursos no mesmo namespace podem ser compartilhados, no entanto, é importante ressaltar que não existe capacidade entre namespaces diferentes. Isso significa que, mesmo que os PVs sejam objetos globais, quando vinculados a um PVC, eles só podem ser acessados por pods que estejam no mesmo namespace. **É fundamental garantir que os namespaces sejam usados para fornecer separação quando apropriado.**

A principal preocupação da maioria das organizações em relação à segurança de dados em um contexto Kubernetes é que um processo em um contêiner possa acessar o armazenamento montado no host, mas que não se destina ao contêiner. "Espaços de nomes" são projetados para evitar esse tipo de comprometimento. No entanto, há uma exceção: contêineres privilegiados.

Um contêiner privilegiado é aquele que é executado com permissões de nível de host substancialmente maiores do que o normal. Essas permissões não são negadas por padrão, portanto, certifique-se de desativar a capacidade usando "políticas de segurança do pod".

Para volumes onde o acesso é desejado tanto do Kubernetes quanto de hosts externos, o armazenamento deve ser gerenciado de maneira tradicional, com o PV introduzido pelo administrador e não gerenciado pelo

Trident. Isso garante que o volume de armazenamento seja destruído somente quando tanto o Kubernetes quanto os hosts externos estiverem desconectados e não estiverem mais utilizando o volume. Além disso, uma política de exportação personalizada pode ser aplicada, permitindo o acesso a partir dos nós do cluster Kubernetes e de servidores direcionados fora do cluster Kubernetes.

Para implantações que possuem nós de infraestrutura dedicados (por exemplo, OpenShift) ou outros nós que não conseguem agendar aplicativos de usuário, políticas de exportação separadas devem ser usadas para limitar ainda mais o acesso aos recursos de storage. Isso inclui a criação de uma política de exportação para serviços que são implantados nesses nós de infraestrutura (por exemplo, os serviços de Métricas e Registro do OpenShift), e para aplicativos padrão que são implantados em nós que não fazem parte da infraestrutura.

Use uma política de exportação dedicada

Você deve garantir que exista uma política de exportação para cada backend que permita acesso apenas aos nós presentes no cluster Kubernetes. Trident pode criar e gerenciar políticas de exportação automaticamente. Dessa forma, Trident limita o acesso aos volumes que provisiona aos nós do cluster Kubernetes e simplifica a adição/remoção de nós.

Alternativamente, você também pode criar uma política de exportação manualmente e preenchê-la com uma ou mais regras de exportação que processem cada solicitação de acesso ao nó:

- Use o comando da CLI `vserver export-policy create` ONTAP para criar a política de exportação.
- Adicione regras à política de exportação usando o `vserver export-policy rule create` comando da CLI do ONTAP.

Executar esses comandos permite que você restrinja quais nós do Kubernetes têm acesso aos dados.

Desative `showmount` para o aplicativo SVM

O `showmount` recurso permite que um cliente NFS consulte a SVM para obter uma lista de exports NFS disponíveis. Um pod implantado no cluster Kubernetes pode executar o `showmount -e` comando contra a SVM e receber uma lista de montagens disponíveis, incluindo aquelas às quais não tem acesso. Embora isso, por si só, não represente uma falha de segurança, fornece informações desnecessárias que podem auxiliar um usuário não autorizado a se conectar a um export NFS.

Você deve desativar `showmount` usando o comando ONTAP CLI em nível de SVM:

```
vserver nfs modify -vserver <svm_name> -showmount disabled
```

Práticas recomendadas do SolidFire

Aprenda as melhores práticas para configurar o armazenamento SolidFire para Trident.

Criar conta SolidFire

Cada conta SolidFire representa um proprietário de volume único e recebe seu próprio conjunto de credenciais do Challenge-Handshake Authentication Protocol (CHAP). Você pode acessar volumes atribuídos a uma conta usando o nome da conta e as credenciais CHAP correspondentes ou por meio de um grupo de acesso a volumes. Uma conta pode ter até dois mil volumes atribuídos a ela, mas um volume pode pertencer a apenas uma conta.

Criar uma política de QoS

Utilize as políticas de qualidade do serviço (QoS) do SolidFire se você quiser criar e salvar uma configuração padronizada de qualidade do serviço que pode ser aplicada a vários volumes.

Você pode definir parâmetros de qualidade do serviço em uma base por volume. O desempenho de cada volume pode ser garantido definindo três parâmetros configuráveis que definem a qualidade do serviço: Min IOPS, Max IOPS e Burst IOPS.

Aqui estão os possíveis valores mínimos, máximos e de burst de IOPS para o tamanho de bloco de 4Kb.

Parâmetro IOPS	Definição	Valor mínimo	Valor padrão	Valor máximo (4Kb)
IOPS mínimo	O nível garantido de desempenho para um volume.	50	50	15000
IOPS máximos	O desempenho não ultrapassará esse limite.	50	15000	200.000
IOPS de rajada	Máximo de IOPS permitido em um cenário de pico curto.	50	15000	200.000

Embora os valores de Max IOPS e Burst IOPS possam ser configurados para até 200.000, o desempenho máximo real de um volume é limitado pelo uso do cluster e pelo desempenho por nó.

O tamanho do bloco e a largura de banda têm influência direta sobre o número de IOPS. À medida que os tamanhos dos blocos aumentam, o sistema aumenta a largura de banda para um nível necessário para processar os blocos maiores. À medida que a largura de banda aumenta, o número de IOPS que o sistema é capaz de atingir diminui. Consulte "Qualidade do serviço do SolidFire" para mais informações sobre QoS e desempenho.

SolidFire autenticação

O Element suporta dois métodos de autenticação: CHAP e Grupos de Acesso a Volumes (VAG). CHAP utiliza o protocolo CHAP para autenticar o host no backend. Grupos de Acesso a Volumes controla o acesso aos volumes que provisiona. NetApp recomenda o uso do CHAP para autenticação, pois é mais simples e não possui limites de escalabilidade.

Trident com o provisionador CSI aprimorado suporta o uso de autenticação CHAP. VAGs devem ser usados apenas no modo de operação tradicional não-CSI.

A autenticação CHAP (verificação de que o iniciador é o usuário pretendido do volume) é suportada apenas com controle de acesso baseado em conta. Se você estiver usando CHAP para autenticação, duas opções estão disponíveis: CHAP unidirecional e CHAP bidirecional. O CHAP unidirecional autentica o acesso ao volume usando o nome da conta SolidFire e o segredo do iniciador. A opção CHAP bidirecional oferece a maneira mais segura de autenticar o volume, pois o volume autentica o host por meio do nome da conta e do segredo do iniciador e, em seguida, o host autentica o volume por meio do nome da conta e do segredo de

destino.

No entanto, se o CHAP não puder ser habilitado e os VAGs forem necessários, crie o grupo de acesso e adicione os iniciadores de host e os volumes ao grupo de acesso. Cada IQN que você adicionar a um grupo de acesso pode acessar cada volume do grupo com ou sem autenticação CHAP. Se o iniciador iSCSI estiver configurado para usar autenticação CHAP, o controle de acesso baseado em conta será usado. Se o iniciador iSCSI não estiver configurado para usar autenticação CHAP, então o controle de acesso do Volume Access Group será usado.

Onde encontrar mais informações?

Algumas das melhores práticas de documentação estão listadas abaixo. Pesquise no "NetApp biblioteca" para as versões mais recentes.

ONTAP

- "NFS Guia de Melhores Práticas e Implementação"
- "Administração SAN" (para iSCSI)
- "Configuração iSCSI Express para RHEL"

Software Element

- "Configurando SolidFire para Linux"

NetApp HCI

- "NetApp HCI pré-requisitos de implantação"
- "Acesse o NetApp Deployment Engine"

Informações sobre as melhores práticas de aplicação

- "Melhores práticas para MySQL no ONTAP"
- "Melhores práticas para MySQL em SolidFire"
- "NetApp SolidFire e Cassandra"
- "Melhores práticas da Oracle sobre SolidFire"
- "Melhores práticas do PostgreSQL em SolidFire"

Nem todas as aplicações possuem diretrizes específicas, é importante trabalhar com sua NetApp equipe e usar o "NetApp biblioteca" para encontrar a documentação mais atualizada.

Integrar Trident

Para integrar Trident, os seguintes elementos de design e arquitetura precisam ser integrados: seleção e implantação de drivers, design de classes de armazenamento, design de pools virtuais, impactos do Persistent Volume Claim (PVC) no provisionamento de storage, operações de volume e implantação de serviços OpenShift usando Trident.

Seleção e implantação de driver

Selecione e implemente um driver de backend para seu sistema de storage.

Drivers de backend ONTAP

Os drivers de backend do ONTAP são diferenciados pelo protocolo utilizado e pela forma como os volumes são provisionados no sistema de storage. Portanto, considere cuidadosamente qual driver implantar.

Em um nível mais alto, se sua aplicação possui componentes que precisam de storage compartilhado (vários pods acessando o mesmo PVC), os drivers baseados em NAS seriam a escolha padrão, enquanto os drivers iSCSI baseados em bloco atendem às necessidades de storage não compartilhado. Escolha o protocolo com base nos requisitos da aplicação e no nível de familiaridade das equipes de storage e infraestrutura. De modo geral, há pouca diferença entre eles para a maioria das aplicações, então, frequentemente, a decisão se baseia na necessidade ou não de storage compartilhado (onde mais de um pod precisará de acesso simultâneo).

Os drivers de backend ONTAP disponíveis são:

- `ontap-nas`: Cada PV provisionado é um ONTAP FlexVolume.
- `ontap-nas-economy`: Cada PV provisionado é uma qtree, com um número configurável de qtrees por FlexVolume (o padrão é 200).
- `ontap-nas-flexgroup`: Cada PV provisionado como um ONTAP FlexGroup, e todos os agregados atribuídos a um SVM são utilizados.
- `ontap-san`: Cada PV provisionado é um LUN dentro de seu próprio FlexVolume.
- `ontap-san-economy`: Cada PV provisionado é um LUN, com um número configurável de LUNs por FlexVolume (o padrão é 100).

A escolha entre os três drivers NAS tem algumas ramificações nos recursos que são disponibilizados para o aplicativo.

Observe que, nas tabelas abaixo, nem todos os recursos são expostos pelo Trident. Alguns devem ser aplicados pelo administrador de storage após o provisionamento, caso essa funcionalidade seja desejada. As notas de rodapé em sobrescrito distinguem a funcionalidade por recurso e driver.

Drivers NAS do ONTAP	Instantâneos	Clones	Políticas de exportação dinâmicas	Multi-attach	QoS	Redimensionar	Replicação
`ontap-nas`	Sim	Sim	Sim nota de rodapé:5[]	Sim	Sim nota de rodapé:1[]	Sim	Sim nota de rodapé:1[]
`ontap-nas-economy`	NO [3]	NO [3]	Sim nota de rodapé:5[]	Sim	NO [3]	Sim	NO [3]
`ontap-nas-flexgroup`	Sim nota de rodapé:1[]	NÃO	Sim nota de rodapé:5[]	Sim	Sim nota de rodapé:1[]	Sim	Sim nota de rodapé:1[]

Trident oferece 2 drivers SAN para ONTAP, cujas capacidades são apresentadas abaixo.

Drivers SAN do ONTAP	Instantâneos	Clones	Multi-attach	CHAP bidirecional	QoS	Redimensionar	Replicação
`ontap-san`	Sim	Sim	Sim	Sim	Sim nota de rodapé:1[]	Sim	Sim nota de rodapé:1[]
`ontap-san-economy`	Sim	Sim	Sim	Sim	NO [3]	Sim	NO [3]

> Nota de rodapé para as tabelas acima: Sim [1]: Não gerenciado pelo Trident Sim [2]: Gerenciado pelo Trident, mas não granular em PV Nãonote:3[]: Não gerenciado pelo Trident e não granular em PV Sim [4]: Suportado para volumes raw-block Sim [5]: Suportado pelo Trident

As funcionalidades que não são granulares em relação ao PV são aplicadas a todo o FlexVolume e todos os PVs (ou seja, qtrees ou LUNs em FlexVols compartilhados) compartilharão um agendamento comum.

Como podemos ver nas tabelas acima, grande parte da funcionalidade entre o `ontap-nas` e o `ontap-nas-economy` é a mesma. No entanto, como o driver `ontap-nas-economy` limita a capacidade de controlar o agendamento na granularidade por PV, isso pode afetar especialmente o seu planejamento de recuperação de desastres e backup. Para equipes de desenvolvimento que desejam aproveitar a funcionalidade de clonagem de PVC no armazenamento ONTAP, isso só é possível ao usar os drivers `ontap-nas`, `ontap-san` ou `ontap-san-economy`.

> O `solidfire-san`driver também é capaz de clonar PVCs.

Drivers de backend do Cloud Volumes ONTAP

Cloud Volumes ONTAP oferece controle de dados juntamente com recursos de armazenamento de nível empresarial para diversos casos de uso, incluindo compartilhamento de arquivos e armazenamento em nível de bloco, atendendo a protocolos NAS e SAN (NFS, SMB / CIFS e iSCSI). Os drivers compatíveis para Cloud Volume ONTAP são `ontap-nas`, `ontap-nas-economy`, `ontap-san` e `ontap-san-economy`. Estes são aplicáveis para Cloud Volume ONTAP para Azure, Cloud Volume ONTAP para GCP.

Drivers de backend do Amazon FSx for ONTAP

Amazon FSx for NetApp ONTAP permite que você aproveite os recursos, o desempenho e as capacidades administrativas do NetApp com os quais você já está familiarizado, enquanto aproveita a simplicidade, agilidade, segurança e escalabilidade de armazenar dados na AWS. FSx for ONTAP oferece suporte a muitos recursos do sistema de arquivos ONTAP e APIs de administração. Os drivers compatíveis para Cloud Volume ONTAP são `ontap-nas`, `ontap-nas-economy`, `ontap-nas-flexgroup`, `ontap-san` e `ontap-san-economy`.

NetApp HCI/SolidFire drivers de backend

O `solidfire-san` driver usado com as plataformas NetApp HCI/SolidFire ajuda o administrador a configurar um backend Element para Trident com base em limites de QoS. Se você deseja projetar seu backend para definir limites de QoS específicos nos volumes provisionados pelo Trident, use o parâmetro `type` no arquivo de backend. O administrador também pode restringir o tamanho do volume que pode ser criado no storage usando o parâmetro `limitVolumeSize`. Atualmente, recursos de storage do Element, como

redimensionamento e replicação de volumes, não são suportados pelo driver `solidfire-san`. Essas operações devem ser realizadas manualmente por meio da interface web do Element Software.

Driver SolidFire	Instantâneos	Clones	Multi-attach	CHAP	QoS	Redimensionar	Replicação
`solidfire-san`	Sim	Sim	Sim	Sim	Sim	Sim	Sim nota de rodapé:1[]

> Nota de rodapé: Simnota de rodapé:1[]: Não gerenciado pelo Trident Simnota de rodapé:2[]: Compatível com volumes de bloco bruto

Drivers de back-end do Azure NetApp Files

Trident usa o `azure-netapp-files` driver para gerenciar o serviço "Azure NetApp Files".

Mais informações sobre este driver e como configurar podem ser encontradas em "Configuração do backend Trident para Azure NetApp Files".

Driver do Azure NetApp Files	Instantâneos	Clones	Multi-attach	QoS	Expandir	Replicação
`azure-netapp-files`	Sim	Sim	Sim	Sim	Sim	Sim nota de rodapé:1[]

> Nota de rodapé: Sim [1]: Não gerenciado por Trident

Design de storage class

Classes de armazenamento individuais precisam ser configuradas e aplicadas para criar um objeto de classe de armazenamento do Kubernetes. Esta seção aborda como projetar uma classe de armazenamento para sua aplicação.

Utilização específica do backend

A filtragem pode ser usada dentro de um objeto de classe de armazenamento específico para determinar qual pool de storage ou conjunto de pools deve ser usado com essa classe de armazenamento específica. Três conjuntos de filtros podem ser definidos na Storage Class: `storagePools`, `additionalStoragePools`, e/ou `excludeStoragePools`.

O `storagePools` parâmetro ajuda a restringir o armazenamento ao conjunto de pools que correspondem a quaisquer atributos especificados. O `additionalStoragePools` parâmetro é usado para estender o conjunto de pools que Trident usa para provisionamento junto com o conjunto de pools selecionados pelos atributos e `storagePools` parâmetros. Você pode usar qualquer um dos parâmetros individualmente ou ambos juntos para garantir que o conjunto apropriado de pools de armazenamento seja selecionado.

O `excludeStoragePools`parâmetro é usado para excluir especificamente o conjunto listado de pools que correspondem aos atributos.

Emular políticas de QoS

Se você deseja projetar classes de armazenamento para emular políticas de Qualidade de Serviço, crie uma classe de armazenamento com o `media` atributo como `hdd` ou `ssd`. Com base no `media` atributo mencionado na storage class, Trident selecionará o backend apropriado que serve `hdd` ou `ssd` agregados para corresponder ao atributo de mídia e então direcionará o provisionamento dos volumes para o agregado específico. Portanto, podemos criar uma storage class PREMIUM que teria o `media` atributo definido como `ssd`, que poderia ser classificada como a política de QoS PREMIUM. Podemos criar outra storage class STANDARD que teria o atributo de mídia definido como `hdd`, que poderia ser classificada como a política de QoS STANDARD. Também podemos usar o atributo `IOPS` na storage class para redirecionar o provisionamento para um appliance Element, que pode ser definido como uma política de QoS.

Utilize backend com base em funcionalidades específicas

As classes de armazenamento podem ser projetadas para direcionar o provisionamento de volumes em um backend específico, onde recursos como provisionamento thin e thick, snapshots, clones e criptografia estão habilitados. Para especificar qual storage usar, crie Storage Classes que especifiquem o backend apropriado com o recurso necessário habilitado.

Pools virtuais

Pools virtuais estão disponíveis para todos os backends do Trident. Você pode definir pools virtuais para qualquer backend, usando qualquer driver que o Trident fornece.

Os pools virtuais permitem que um administrador crie um nível de abstração sobre backends que podem ser referenciados por meio de Storage Classes, para maior flexibilidade e alocação eficiente de volumes nos backends. Diferentes backends podem ser definidos com a mesma classe de serviço. Além disso, vários storage pools podem ser criados no mesmo backend, mas com características diferentes. Quando uma Storage Class é configurada com um seletor com rótulos específicos, Trident escolhe um backend que corresponda a todos os rótulos do seletor para alocar o volume. Se os rótulos do seletor da Storage Class corresponderem a vários storage pools, Trident escolherá um deles para provisionar o volume.

Design de pool virtual

Ao criar um backend, geralmente é possível especificar um conjunto de parâmetros. Era impossível para o administrador criar outro backend com as mesmas credenciais de storage e com um conjunto diferente de parâmetros. Com a introdução dos pools virtuais, esse problema foi atenuado. Um pool virtual é uma camada de abstração introduzida entre o backend e a Kubernetes Storage Class, permitindo que o administrador defina parâmetros juntamente com rótulos que podem ser referenciados por meio das Kubernetes Storage Classes como um seletor, de forma agnóstica ao backend. Pools virtuais podem ser definidos para todos os backends NetApp compatíveis com Trident. Essa lista inclui SolidFire/NetApp HCI, ONTAP, assim como Azure NetApp Files.

> Ao definir pools virtuais, recomenda-se não tentar reorganizar a ordem dos pools virtuais existentes em uma definição de backend. Também é aconselhável não editar/modificar atributos de um pool virtual existente e definir um novo pool virtual em vez disso.

Emulando diferentes níveis de serviço/QoS

É possível projetar pools virtuais para emular classes de serviço. Usando a implementação de pool virtual para Cloud Volume Service for Azure NetApp Files, vamos examinar como podemos configurar diferentes classes de serviço. Configure o backend do Azure NetApp Files com vários rótulos, representando diferentes níveis de desempenho. Defina o aspecto `servicelevel` para o nível de desempenho apropriado e adicione outros aspectos necessários sob cada rótulo. Agora crie diferentes Storage Classes do Kubernetes que serão

mapeadas para diferentes pools virtuais. Usando o campo `parameters.selector`, cada StorageClass especifica quais pools virtuais podem ser usados para hospedar um volume.

Atribuindo conjunto específico de aspectos

É possível projetar vários pools virtuais com um conjunto específico de aspectos a partir de um único backend de armazenamento. Para fazer isso, configure o backend com vários rótulos e defina os aspectos necessários em cada rótulo. Agora crie diferentes classes de armazenamento do Kubernetes usando o `parameters.selector` campo que mapeará para diferentes pools virtuais. Os volumes provisionados no backend terão os aspectos definidos no pool virtual escolhido.

Características do PVC que afetam o provisionamento de storage

Alguns parâmetros além da classe de armazenamento solicitada podem afetar o processo de decisão de provisionamento do Trident ao criar um PVC.

Modo de acesso

Ao solicitar armazenamento por meio de um PVC, um dos campos obrigatórios é o modo de acesso. O modo desejado pode afetar o backend selecionado para hospedar a solicitação de armazenamento.

Trident tentará encontrar uma correspondência entre o protocolo de storage utilizado e o método de acesso especificado, de acordo com a seguinte matriz. Isso é independente da plataforma de storage subjacente.

	ReadWriteOnce	ReadOnlyMany	ReadWriteMany
iSCSI	Sim	Sim	Sim (bloco bruto)
NFS	Sim	Sim	Sim

Uma solicitação de um ReadWriteMany PVC enviada a uma implantação do Trident sem um backend NFS configurado resultará em nenhum volume sendo provisionado. Por esse motivo, o solicitante deve usar o modo de acesso apropriado para sua aplicação.

Operações de volume

Modificar volumes persistentes

Os volumes persistentes são, com duas exceções, objetos imutáveis no Kubernetes. Uma vez criados, a política de recuperação e o tamanho podem ser modificados. No entanto, isso não impede que alguns aspectos do volume sejam modificados fora do Kubernetes. Isso pode ser desejável para personalizar o volume para aplicações específicas, garantir que a capacidade não seja consumida acidentalmente ou simplesmente mover o volume para um controlador de storage diferente por qualquer motivo.

> Os provisionadores integrados do Kubernetes não suportam operações de redimensionamento de volumes para NFS, iSCSI ou FC PVs neste momento. Trident suporta a expansão de volumes NFS, iSCSI e FC.

Os detalhes de conexão do PV não podem ser modificados após a criação.

Criar snapshots de volume sob demanda

Trident suporta a criação de snapshots de volumes sob demanda e a criação de PVCs a partir de snapshots usando a estrutura CSI. Os snapshots fornecem um método conveniente para manter cópias dos dados em

um ponto no tempo e têm um ciclo de vida independente do PV de origem no Kubernetes. Esses snapshots podem ser usados para clonar PVCs.

Criar volumes a partir de snapshots

Trident também suporta a criação de PersistentVolumes a partir de snapshots de volume. Para isso, basta criar um PersistentVolumeClaim e mencionar o `datasource` como o snapshot necessário a partir do qual o volume precisa ser criado. Trident gerenciará esse PVC criando um volume com os dados presentes no snapshot. Com esse recurso, é possível duplicar dados entre regiões, criar ambientes de teste, substituir um volume de produção danificado ou corrompido por completo ou recuperar arquivos e diretórios específicos e transferi-los para outro volume conectado.

Mover volumes no cluster

Os administradores de storage têm a capacidade de mover volumes entre agregados e controladores no cluster ONTAP de forma não disruptiva para o consumidor de storage. Essa operação não afeta o Trident nem o cluster Kubernetes, desde que o agregado de destino seja um ao qual a SVM que o Trident está utilizando tenha acesso. É importante ressaltar que, se o agregado foi adicionado recentemente à SVM, o backend precisará ser atualizado ao ser readicionado ao Trident. Isso fará com que o Trident faça um novo inventário da SVM para que o novo agregado seja reconhecido.

No entanto, a movimentação de volumes entre backends não é suportada automaticamente pelo Trident. Isso inclui entre SVMs no mesmo cluster, entre clusters ou para uma plataforma de storage diferente (mesmo que esse sistema de storage esteja conectado ao Trident).

Se um volume for copiado para outro local, o recurso de importação de volume pode ser usado para importar volumes atuais para Trident.

Expandir volumes

Trident suporta o redimensionamento de PVs NFS, iSCSI e FC. Isso permite que os usuários redimensionem seus volumes diretamente pela camada Kubernetes. A expansão de volumes é possível para todas as principais plataformas de storage NetApp, incluindo ONTAP, e backends SolidFire/NetApp HCI. Para permitir uma possível expansão futura, defina `allowVolumeExpansion` como `true` no seu StorageClass associado ao volume. Sempre que o Persistent Volume precisar ser redimensionado, edite a anotação `spec.resources.requests.storage` no Persistent Volume Claim para o tamanho de volume necessário. Trident cuidará automaticamente do redimensionamento do volume no cluster de storage.

Importar um volume existente ao Kubernetes

A importação de volumes permite importar um volume de armazenamento existente para um ambiente Kubernetes. Atualmente, isso é suportado pelos `ontap-nas`, `ontap-nas-flexgroup`, `solidfire-san` e `azure-netapp-files` drivers. Esse recurso é útil ao migrar um aplicativo existente para o Kubernetes ou em cenários de recuperação de desastres.

Ao usar os drivers ONTAP e `solidfire-san`, utilize o comando `tridentctl import volume <backend-name> <volume-name> -f /path/pvc.yaml` para importar um volume existente para o Kubernetes para ser gerenciado pelo Trident. O arquivo PVC YAML ou JSON usado no comando de importação de volume aponta para uma storage class que identifica o Trident como o provisionador. Ao usar um backend NetApp HCI/SolidFire, certifique-se de que os nomes dos volumes sejam únicos. Se os nomes dos volumes estiverem duplicados, clone o volume para um nome único para que o recurso de importação de volume possa distingui-los.

Se o `azure-netapp-files` driver for utilizado, use o comando `tridentctl import volume`

`<backend-name> <volume path> -f /path/pvc.yaml` para importar o volume para o Kubernetes para ser gerenciado pelo Trident. Isso garante uma referência de volume exclusiva.

Ao executar o comando acima, Trident localizará o volume no backend e lerá seu tamanho. Ele adicionará automaticamente (e sobrescreverá, se necessário) o tamanho do volume do PVC configurado. Em seguida, Trident cria o novo PV e o Kubernetes vincula o PVC ao PV.

Se um contêiner foi implantado de forma que exigisse o PVC importado específico, ele permanecerá em estado pendente até que o par PVC/PV seja vinculado por meio do processo de importação de volume. Após a vinculação do par PVC/PV, o contêiner deverá ser iniciado, desde que não haja outros problemas.

Serviço de registro

A implantação e o gerenciamento do armazenamento para o registro foram documentados em "netapp.io" no "blog".

Serviço de logging

Assim como outros OpenShift serviços, o serviço de registro de logs é implantado usando o Ansible com parâmetros de configuração fornecidos pelo arquivo de inventário, também conhecido como hosts, fornecido ao playbook. Há dois métodos de instalação que serão abordados: implantar o serviço de registro de logs durante a instalação inicial do OpenShift e implantar o serviço de registro de logs após o OpenShift ter sido instalado.

> A partir da versão 3.9 do Red Hat OpenShift, a documentação oficial recomenda não utilizar NFS para o serviço de registro de logs devido a preocupações com corrupção de dados. Isso se baseia em testes realizados pela Red Hat em seus produtos. O servidor NFS do ONTAP não apresenta esses problemas e pode facilmente suportar uma implementação de registro de logs. Em última análise, a escolha do protocolo para o serviço de registro de logs é sua, apenas saiba que ambos funcionarão muito bem ao usar plataformas NetApp e não há motivo para evitar NFS se essa for sua preferência.

Se optar por usar NFS com o serviço de registro de logs, você precisará definir a variável Ansible `openshift_enable_unsupported_configurations` para `true` evitar que o instalador falhe.

Comece agora

O serviço de registro de logs pode, opcionalmente, ser implementado tanto para aplicações quanto para as operações principais do próprio cluster OpenShift. Se você optar por implementar o registro de logs de operações, especificando a variável `openshift_logging_use_ops` como `true`, duas instâncias do serviço serão criadas. As variáveis que controlam a instância de registro de logs para operações contêm "ops", enquanto a instância para aplicações não.

Configurar as variáveis do Ansible de acordo com o método de implantação é importante para garantir que o storage correto seja utilizado pelos serviços subjacentes. Vamos analisar as opções para cada um dos métodos de implantação.

> As tabelas abaixo contêm apenas as variáveis relevantes para configuração de storage conforme relacionado ao serviço de registro de logs. Você pode encontrar outras opções em "Documentação de registro de logs do Red Hat OpenShift" que devem ser revisadas, configuradas e utilizadas de acordo com a sua implementação.

As variáveis na tabela abaixo farão com que o playbook do Ansible crie um PV e um PVC para o serviço de logging usando os detalhes fornecidos. Este método é significativamente menos flexível do que usar o

playbook de instalação de componentes após a instalação do OpenShift, porém, se você já tiver volumes disponíveis, é uma opção.

Variável	Detalhes
`openshift_logging_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de registro de logs.
`openshift_logging_storage_host`	O nome do host ou endereço IP do host NFS. Isso deve ser definido como o dataLIF da sua máquina virtual.
`openshift_logging_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume estiver vinculado como `/openshift_logging`, você usaria esse caminho para esta variável.
`openshift_logging_storage_volume_name`	O nome, por exemplo, `pv_ose_logs`, do PV a ser criado.
`openshift_logging_storage_volume_size`	O tamanho da exportação NFS, por exemplo `100Gi`.

Se o seu OpenShift cluster já estiver em execução e, portanto, Trident já tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As seguintes variáveis precisarão ser configuradas.

Variável	Detalhes
`openshift_logging_es_pvc_dynamic`	Defina como true para usar volumes provisionados dinamicamente.
`openshift_logging_es_pvc_storage_class_name`	O nome da storage class que será utilizada no PVC.
`openshift_logging_es_pvc_size`	O tamanho do volume solicitado no PVC.
`openshift_logging_es_pvc_prefix`	Um prefixo para os PVCs usados pelo serviço de logging.
`openshift_logging_es_ops_pvc_dynamic`	Defina como `true` para usar volumes provisionados dinamicamente para a instância de logging de operações.
`openshift_logging_es_ops_pvc_storage_class_name`	O nome da storage class para a instância de ops logging.
`openshift_logging_es_ops_pvc_size`	O tamanho da solicitação de volume para a instância ops.
`openshift_logging_es_ops_pvc_prefix`	Um prefixo para os PVCs da instância ops.

Implante a pilha de registro de logs

Se você estiver implementando o registro de logs como parte do processo de instalação inicial do OpenShift, basta seguir o processo de implantação padrão. O Ansible configurará e implantará os serviços e objetos do OpenShift necessários para que o serviço esteja disponível assim que o Ansible for concluído.

No entanto, se você estiver realizando a implantação após a instalação inicial, o playbook do componente precisará ser usado pelo Ansible. Esse processo pode variar um pouco com diferentes versões do OpenShift,

portanto, certifique-se de ler e seguir "Documentação do Red Hat OpenShift Container Platform 3.11" para a sua versão.

Serviço de métricas

O serviço de métricas fornece informações valiosas ao administrador sobre o status, a utilização de recursos e a disponibilidade do OpenShift cluster. Ele também é necessário para a funcionalidade de escalonamento automático de pods e muitas organizações utilizam dados do serviço de métricas para seus aplicativos de cobrança e/ou demonstração de custos.

Assim como no serviço de logging, e OpenShift como um todo, Ansible é usado para implantar o serviço de métricas. Também, como o serviço de logging, o serviço de métricas pode ser implantado durante a configuração inicial do cluster ou após sua operação, utilizando o método de instalação de componentes. As tabelas a seguir contêm as variáveis que são importantes ao configurar storage persistente para o serviço de métricas.

As tabelas abaixo contêm apenas as variáveis relevantes para configuração de storage conforme relacionado ao serviço de métricas. Existem muitas outras opções encontradas na documentação que devem ser revisadas, configuradas e utilizadas de acordo com a sua implementação.

Variável	Detalhes
`openshift_metrics_storage_kind`	Defina como `nfs` para que o instalador crie um NFS PV para o serviço de registro de logs.
`openshift_metrics_storage_host`	O nome do host ou endereço IP do host NFS. Este valor deve ser definido como o dataLIF da sua SVM.
`openshift_metrics_storage_nfs_directory`	O caminho de montagem para a exportação NFS. Por exemplo, se o volume estiver vinculado como `/openshift_metrics`, você usaria esse caminho para esta variável.
`openshift_metrics_storage_volume_name`	O nome, por exemplo, `pv_ose_metrics`, do PV a ser criado.
`openshift_metrics_storage_volume_size`	O tamanho da exportação NFS, por exemplo `100Gi`.

Se o seu OpenShift cluster já estiver em execução e, portanto, Trident já tiver sido implantado e configurado, o instalador poderá usar o provisionamento dinâmico para criar os volumes. As seguintes variáveis precisarão ser configuradas.

Variável	Detalhes
`openshift_metrics_cassandra_pvc_prefix`	Um prefixo a ser usado para os PVCs de métricas.
`openshift_metrics_cassandra_pvc_size`	O tamanho dos volumes a serem solicitados.
`openshift_metrics_cassandra_storage_type`	O tipo de storage a ser usado para métricas, isso deve ser definido como dinâmico para que o Ansible crie PVCs com a classe de storage apropriada.
`openshift_metrics_cassanda_pvc_storage_class_name`	O nome da storage class a ser usada.

Implante o serviço de métricas

Com as variáveis Ansible apropriadas definidas no seu arquivo hosts/inventory, implante o serviço usando Ansible. Se você estiver implantando no momento da instalação do OpenShift, o PV será criado e usado automaticamente. Se estiver implantando usando os playbooks de componentes, após a instalação do OpenShift, o Ansible criará todos os PVCs necessários e, depois que Trident provisionar storage para eles, implantará o serviço.

As variáveis acima e o processo de implantação podem mudar a cada versão do OpenShift. Certifique-se de revisar e seguir "Guia de implantação da Red Hat's OpenShift" para a sua versão para que esteja configurado para o seu ambiente.

Proteção de dados e recuperação de desastres

Saiba mais sobre as opções de proteção e recuperação para Trident e volumes criados usando Trident. Você deve ter uma estratégia de proteção e recuperação de dados para cada aplicação com requisito de persistência.

Replicação e recuperação do Trident

Você pode criar um backup para restaurar Trident em caso de desastre.

Trident replicação

Trident usa CRDs do Kubernetes para armazenar e gerenciar seu próprio estado e o etcd do cluster Kubernetes para armazenar seus metadados.

Passos

1. Faça backup do cluster Kubernetes etcd usando "Kubernetes: fazendo backup de um cluster etcd".
2. Coloque os artefatos de backup em um FlexVol volume

> NetApp recomenda que você proteja a SVM onde o FlexVol reside com uma relação de SnapMirror para outra SVM.

Recuperação do Trident

Utilizando CRDs do Kubernetes e o snapshot do etcd do cluster Kubernetes, você pode recuperar Trident.

Passos

1. A partir da SVM de destino, monte o volume que contém os arquivos de dados e certificados do Kubernetes etcd no host que será configurado como nó mestre.
2. Copie todos os certificados necessários referentes ao cluster Kubernetes em `/etc/kubernetes/pki` e os arquivos de membros do etcd em `/var/lib/etcd`.
3. Restaure o cluster Kubernetes a partir do backup etcd usando "Kubernetes: restaurando um cluster etcd".
4. Execute `kubectl get crd` para verificar se todos os recursos personalizados do Trident foram iniciados e recupere os objetos do Trident para verificar se todos os dados estão disponíveis.

Replicação e recuperação de SVM

Trident não consegue configurar relações de replicação, no entanto, o administrador de storage pode usar "ONTAP SnapMirror" para replicar uma SVM.

Em caso de desastre, você pode ativar a SVM de destino do SnapMirror para começar a fornecer dados. Você pode retornar à primária quando os sistemas forem restaurados.

Sobre esta tarefa

Considere o seguinte ao usar o recurso de Replicação SVM do SnapMirror:

- Você deve criar um backend distinto para cada SVM com SVM-DR habilitado.
- Configure as classes de armazenamento para selecionar apenas os backends replicados quando necessário, para evitar que volumes que não precisam de replicação sejam provisionados nos backends que suportam SVM-DR.
- Os administradores de aplicativos devem compreender os custos e a complexidade adicionais associados à replicação e considerar cuidadosamente seu plano de recuperação antes de iniciar este processo.

Replicação SVM

Você pode usar "ONTAP: SnapMirror SVM replicação" para criar a relação de replicação SVM.

SnapMirror permite definir opções para controlar o que replicar. Você precisará saber quais opções selecionou ao executar Recuperação de SVM usando Trident.

- "-identity-preserve true" replica toda a configuração da SVM.
- "-descartar-configs network" Exclui LIFs e configurações de rede relacionadas.
- "-identity-preserve false" replica apenas os volumes e a configuração de segurança.

Recuperação de SVM usando Trident

Trident não detecta automaticamente falhas de SVM. Em caso de desastre, o administrador pode iniciar manualmente o failover do Trident para a nova SVM.

Passos

1. Cancele as transferências SnapMirror agendadas e em andamento, interrompa a relação de replicação, pare o SVM de origem e, em seguida, ative o SVM de destino SnapMirror.
2. Se você especificou `-identity-preserve false` ou `-discard-config network` ao configurar a replicação do SVM, atualize o `managementLIF` e o `dataLIF` no arquivo de definição do backend do Trident.
3. Confirme `storagePrefix` está presente no arquivo de definição do backend Trident. Este parâmetro não pode ser alterado. Omitir `storagePrefix` fará com que a atualização do backend falhe.
4. Atualize todos os backends necessários para refletir o novo nome da SVM de destino usando:

```
./tridentctl update backend <backend-name> -f <backend-json-file> -n
<namespace>
```

5. Se você especificou `-identity-preserve false` ou `discard-config network`, você deve reiniciar todos os pods do aplicativo.

Se você especificou `-identity-preserve true`, todos os volumes provisionados pelo Trident começam a fornecer dados quando o SVM de destino é ativado.

Replicação e recuperação de volume

Trident não consegue configurar relações de replicação do SnapMirror, no entanto, o administrador de storage pode usar "ONTAP SnapMirror replicação e recuperação" para replicar volumes criados pelo Trident.

Em seguida, você pode importar os volumes recuperados para Trident usando "tridentctl volume import".

A importação não é suportada em `ontap-nas-economy`, `ontap-san-economy` ou `ontap-flexgroup-economy` drivers.

Proteção de dados Snapshot

Você pode proteger e restaurar dados usando:

- Um controlador de snapshot externo e CRDs para criar snapshots de Persistent Volumes (PVs) do Kubernetes.

  "Instantâneos de volume"

- ONTAP Snapshots para restaurar todo o conteúdo de um volume ou recuperar arquivos ou LUNs individuais.

  "Snapshots do ONTAP"

Automatizando o failover de aplicações stateful com Trident

O recurso de desanexação forçada do Trident permite desanexar automaticamente volumes de nós com problemas em um cluster Kubernetes, evitando corrupção de dados e garantindo a disponibilidade do aplicativo. Esse recurso é particularmente útil em cenários onde os nós ficam inativos ou são colocados offline para manutenção.

Detalhes sobre force detach

A desanexação forçada está disponível para `ontap-san`, `ontap-san-economy`, `ontap-nas` e `ontap-nas-economy` apenas. Antes de habilitar a desanexação forçada, o desligamento não gradual do nó (NGNS) deve estar habilitado no cluster Kubernetes. O NGNS está habilitado por padrão para Kubernetes 1.28 e superiores. Para mais informações, consulte "Kubernetes: encerramento abrupto de nós".

Ao usar o `ontap-nas` ou `ontap-nas-economy` driver, você precisa definir o parâmetro `autoExportPolicy` na configuração do backend para `true` que o Trident possa restringir o acesso do nó Kubernetes com o taint aplicado usando políticas de exportação gerenciadas.

Como Trident depende do NGNS do Kubernetes, não remova `out-of-service` taints de um nó não íntegro até que todas as cargas de trabalho não toleráveis sejam reagendadas. Aplicar ou remover a taint de forma imprudente pode comprometer a proteção de dados do backend.

Quando o administrador do cluster Kubernetes aplicou a `node.kubernetes.io/out-of-service=nodeshutdown:NoExecute` taint ao nó e `enableForceDetach` está definida como `true`, Trident determinará o status do nó e:

1. Interrompa o acesso de E/S de backend para volumes montados nesse nó.
2. Marque o objeto do nó Trident como `dirty` (não seguro para novas publicações).

> O controlador Trident rejeitará novas solicitações de publicação de volume até que o nó seja requalificado (após ter sido marcado como `dirty`) pelo pod do nó Trident. Quaisquer cargas de trabalho agendadas com um PVC montado (mesmo depois que o nó de cluster estiver íntegro e pronto) não serão aceitas até que Trident possa verificar o nó `clean` (seguro para novas publicações).

Quando a integridade do nó for restaurada e a taint for removida, Trident irá:

1. Identifique e limpe caminhos publicados obsoletos no nó.
2. Se o nó estiver em um `cleanable` estado (a marcação de fora de serviço foi removida e o nó está em `Ready` estado) e todos os caminhos publicados obsoletos estiverem limpos, Trident readmitirá o nó como `clean` e permitirá novos volumes publicados no nó.

Detalhes sobre failover automatizado

Você pode automatizar o processo de desanexação forçada por meio da integração com "operador de verificação de integridade do nó (NHC)". Quando ocorre uma falha em um nó, o NHC aciona a remediação de nó do Trident (TNR) e a desanexação forçada automaticamente, criando um CR TridentNodeRemediation no namespace do Trident definindo o nó com falha. O TNR é criado somente após a falha do nó e removido pelo NHC assim que o nó volta a ficar online ou é excluído.

Falha no processo de remoção do pod do nó

O failover automatizado seleciona as cargas de trabalho a serem removidas do nó com falha. Quando um TNR é criado, o controlador TNR marca o nó como sujo, impedindo novas publicações de volume e começa a remover os pods compatíveis com desanexação forçada e seus anexos de volume.

Todos os volumes/PVCs suportados por force-detach são suportados por failover automatizado:

- NAS, e volumes NAS-economy usando políticas de exportação automática (SMB ainda não é suportado).
- SAN e volumes SAN-economy.

Consulte Detalhes sobre force detach.

Comportamento padrão:

- Os pods que utilizam volumes compatíveis com o recurso de desanexação forçada são removidos do nó com falha. O Kubernetes irá reagendá-los para um nó íntegro.
- Os pods que utilizam um volume não suportado pelo force-detach, incluindo volumes não-Trident, não são removidos do nó com falha.
- Os pods sem estado (não PVCs) não são removidos do nó com falha, a menos que a anotação do pod `trident.netapp.io/podRemediationPolicy: delete` esteja definida.

Substituindo o comportamento de remoção do pod:

O comportamento de remoção de pods pode ser personalizado usando uma anotação de pod: `trident.netapp.io/podRemediationPolicy[retain, delete]`. Essas anotações são examinadas e usadas quando ocorre um failover. Aplique anotações à especificação do pod do deployment/replicaset do Kubernetes para evitar que a anotação desapareça após um failover:

- `retain` - O pod NÃO será removido do nó com falha durante um failover automatizado.
- `delete` - O Pod será removido do nó com falha durante um failover automatizado.

Essas anotações podem ser aplicadas a qualquer pod.

- As operações de I/O serão bloqueadas apenas em nós com falha para volumes que suportam force-detach.
- Para volumes que não suportam force-detach, existe o risco de corrupção de dados e problemas de multi-attach.

CR TridentNodeRemediation

O TridentNodeRemediation (TNR) CR define um nó com falha. O nome do TNR é o nome do nó com falha.

Exemplo de TNR:

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediation
metadata:
  name: <K8s-node-name>
spec: {}
```

TNR states: Use os seguintes comandos para visualizar o status dos TNRs:
`kubectl get tnr <name> -n <trident-namespace>`

Os TNRs podem estar em um dos seguintes estados:

- *Remediando*:
  - Cessar o acesso de E/S de backend para volumes suportados por force-detach montados nesse nó.
  - O objeto de nó Trident está marcado como sujo (não é seguro para novas publicações).
  - Remova os pods e os anexos de volume do nó
- *Recuperação de nó pendente*:
  - O controlador está aguardando que o nó volte a ficar online.
  - Assim que o nó estiver online, publish-enforcement garantirá que o nó esteja limpo e pronto para novas publicações de volume.
- Se o nó for excluído do K8s, o controlador TNR removerá o TNR e interromperá a reconciliação.
- *Concluído*:
  - Todas as etapas de remediação e recuperação do nó foram concluídas com sucesso. O nó está limpo e pronto para novas publicações de volume.
- *Fracassado*:

- Erro irrecuperável. Os motivos do erro são definidos no campo status.message do CR.

Habilitando o failover automatizado

Pré-requisitos:

- Certifique-se de que o recurso de desanexação forçada esteja ativado antes de ativar o failover automatizado. Para mais informações, consulte Detalhes sobre force detach.
- Instale a verificação de integridade (NHC) no cluster Kubernetes.
  - "Instalar o operator-sdk".
  - Instale o Operator Lifecycle Manager (OLM) no cluster se ainda não estiver instalado: `operator-sdk olm install`.
  - Instale o operador de verificação de integridade do nó: `kubectl create -f https://operatorhub.io/install/node-healthcheck-operator.yaml`.

> Você também pode usar métodos alternativos para detectar falha de nó, conforme especificado na [Integrating Custom Node Health Check Solutions] seção abaixo.

Consulte "Operador de verificação de integridade do nó" para mais informações.

Passos

1. Crie um NodeHealthCheck (NHC) no namespace Trident para monitorar os nós de trabalho no cluster. Exemplo:

```
apiVersion: remediation.medik8s.io/v1alpha1
kind: NodeHealthCheck
metadata:
  name: <CR name>
spec:
  selector:
    matchExpressions:
      - key: node-role.kubernetes.io/control-plane
        operator: DoesNotExist
      - key: node-role.kubernetes.io/master
        operator: DoesNotExist
  remediationTemplate:
    apiVersion: trident.netapp.io/v1
    kind: TridentNodeRemediationTemplate
    namespace: <Trident installation namespace>
    name: trident-node-remediation-template
  minHealthy: 0 # Trigger force-detach upon one or more node failures
  unhealthyConditions:
    - type: Ready
      status: "False"
      duration: 0s
    - type: Ready
      status: Unknown
      duration: 0s
```

2. Aplique o CR de verificação de integridade do nó no `trident` namespace.

```
kubectl apply -f <nhc-cr-file>.yaml -n <trident-namespace>
```

A configuração CR acima monitora os nós de trabalho do K8s em busca das condições de nó Ready: false e Unknown. O failover automatizado será acionado quando um nó entrar no estado Ready: false ou Ready: Unknown.

O `unhealthyConditions` no CR usa um período de carência de 0 segundos. Isso faz com que o failover automatizado seja acionado imediatamente após o K8s definir a condição do nó como Ready: false, que é definida depois que o K8s perde o heartbeat de um nó. O K8s tem um padrão de espera de 40 segundos após o último heartbeat antes de definir Ready: false. Esse período de carência pode ser personalizado nas opções de implantação do K8s.

Para opções de configuração adicionais, consulte "Documentação do Node-Healthcheck-Operator".

Informações adicionais de configuração

Quando Trident é instalado com o recurso de desanexação forçada ativado, dois recursos adicionais são criados automaticamente no namespace do Trident para facilitar a integração com NHC: TridentNodeRemediationTemplate (TNRT) e ClusterRole.

TridentNodeRemediationTemplate (TNRT):

O TNRT serve como modelo para o controlador NHC, que usa TNRT para gerar recursos TNR conforme necessário.

```
apiVersion: trident.netapp.io/v1
kind: TridentNodeRemediationTemplate
metadata:
  name: trident-node-remediation-template
  namespace: trident
spec:
  template:
    spec: {}
```

ClusterRole:

Uma função de cluster também é adicionada durante a instalação quando force-detach está habilitado. Isso concede permissões ao NHC para os TNRs no namespace Trident.

```
apiVersion: rbac.authorization.k8s.io/v1
kind: ClusterRole
metadata:
  labels:
    rbac.ext-remediation/aggregate-to-ext-remediation: "true"
  name: tridentnoderemediation-access
rules:
- apiGroups:
  - trident.netapp.io
  resources:
  - tridentnoderemediationtemplates
  - tridentnoderemediations
  verbs:
  - get
  - list
  - watch
  - create
  - update
  - patch
  - delete
```

Atualizações e manutenção de cluster K8s

Para evitar qualquer failover, pause o failover automatizado durante a manutenção ou atualizações do K8s, quando se espera que os nós fiquem indisponíveis ou reinicializem. Você pode pausar o CR do NHC (descrito acima) aplicando um patch no seu CR:

```
kubectl patch NodeHealthCheck <cr-name> --patch
'{"spec":{"pauseRequests":["<description-for-reason-of-pause>"]}}' --type=merge
```

Isso pausa o failover automatizado. Para reativar o failover automatizado, remova o pauseRequests da especificação após a conclusão da manutenção.

Limitações

- As operações de E/S são bloqueadas apenas nos nós com falha para volumes compatíveis com force-detach. Somente os pods que utilizam volumes/PVCs compatíveis com force-detach são removidos automaticamente.
- O failover automatizado e o force-detach são executados dentro do pod trident-controller. Se o nó que hospeda o trident-controller falhar, o failover automatizado será atrasado até que o K8s mova o pod para um nó saudável.

Integrando soluções personalizadas de verificação de integridade de nós

Você pode substituir o Node Healthcheck Operator por ferramentas alternativas de detecção de falhas de nó para acionar o failover automatizado. Para garantir a compatibilidade com o mecanismo de failover automatizado, sua solução personalizada deve:

- Crie um TNR quando uma falha de nó for detectada, usando o nome do nó com falha como o nome do CR do TNR.
- Exclua o TNR quando o nó se recuperar e o TNR estiver no estado Succeeded.

Segurança

Segurança

Utilize as recomendações listadas aqui para garantir que sua instalação do Trident seja segura.

Execute Trident em seu próprio namespace

É importante impedir que aplicativos, administradores de aplicativos, usuários e aplicativos de gerenciamento acessem as definições de objetos do Trident ou os pods para garantir um armazenamento confiável e bloquear possíveis atividades maliciosas.

Para separar os outros aplicativos e usuários do Trident, sempre instale Trident em seu próprio namespace do Kubernetes (`trident`). Colocar Trident em seu próprio namespace garante que apenas a equipe administrativa do Kubernetes tenha acesso ao pod do Trident e aos artefatos (como segredos de backend e CHAP, se aplicável) armazenados nos objetos CRD do namespace. Você deve garantir que somente os administradores tenham acesso ao namespace do Trident e, assim, acesso ao `tridentctl` aplicativo.

Use autenticação CHAP com backends SAN do ONTAP

Trident oferece suporte à autenticação baseada em CHAP para cargas de trabalho ONTAP SAN (usando os drivers `ontap-san` e `ontap-san-economy`). NetApp recomenda o uso de CHAP bidirecional com Trident para autenticação entre um host e o backend de storage.

Para backends ONTAP que utilizam os drivers de armazenamento SAN, Trident pode configurar o CHAP bidirecional e gerenciar nomes de usuário e segredos CHAP por meio de `tridentctl`. Consulte "Prepare-se para configurar o backend com os drivers ONTAP SAN" para entender como o Trident configura o CHAP em backends ONTAP.

Use autenticação CHAP com NetApp HCI e SolidFire backends

NetApp recomenda implantar CHAP bidirecional para garantir autenticação entre um host e os backends NetApp HCI e SolidFire. Trident usa um objeto secreto que inclui duas senhas CHAP por tenant. Quando Trident é instalado, ele gerencia os segredos CHAP e os armazena em um `tridentvolume` objeto CR para o respectivo PV. Quando você cria um PV, Trident usa os segredos CHAP para iniciar uma sessão iSCSI e se comunicar com o sistema NetApp HCI e SolidFire via CHAP.

Os volumes criados pelo Trident não estão associados a nenhum Volume Access Group.

Use Trident com NVE e NAE

NetApp ONTAP fornece criptografia de dados em repouso para proteger dados confidenciais caso um disco seja roubado, devolvido ou reutilizado. Para detalhes, consulte "Configurar visão geral da criptografia de volume NetApp".

- Se o NAE estiver habilitado no backend, qualquer volume provisionado no Trident será habilitado para NAE.
  - Você pode definir o sinalizador de criptografia NVE para `""` criar volumes com NAE habilitado.
- Se o NAE não estiver habilitado no backend, qualquer volume provisionado no Trident terá o NVE habilitado, a menos que o sinalizador de criptografia NVE esteja definido como `false` (o valor padrão) na configuração do backend.

Volumes criados no Trident em um backend habilitado para NAE devem ser criptografados com NVE ou NAE.

- Você pode definir o sinalizador de criptografia NVE `true` na configuração do backend Trident para substituir a criptografia NAE e usar uma chave de criptografia específica para cada volume.
- Definir o sinalizador de criptografia NVE para `false` em um backend com NAE habilitado cria um volume com NAE habilitado. Você não pode desabilitar a criptografia NAE definindo o sinalizador de criptografia NVE para `false`.

- Você pode criar manualmente um volume NVE no Trident definindo explicitamente o sinalizador de criptografia NVE como `true`.

Para obter mais informações sobre as opções de configuração do backend, consulte:

- "Opções de configuração do ONTAP SAN"
- "Opções de configuração do ONTAP NAS"

Linux Unified Key Setup (LUKS)

Você pode habilitar o Linux Unified Key Setup (LUKS) para criptografar volumes ONTAP SAN e ONTAP SAN ECONOMY no Trident. Trident oferece suporte à rotação de senha e expansão de volume para volumes criptografados com LUKS.

No Trident, os volumes criptografados com LUKS usam a cifra e o modo aes-xts-plain64, conforme recomendado por "NIST".

A criptografia LUKS não é compatível com sistemas ASA r2. Para obter informações sobre sistemas ASA r2, consulte "Saiba mais sobre os sistemas de armazenamento ASA r2".

Antes de começar

- Os nós de trabalho devem ter cryptsetup 2.1 ou superior (mas inferior a 3.0) instalado. Para mais informações, visite "Gitlab: cryptsetup".
- Por motivos de desempenho, NetApp recomenda que os nós de trabalho suportem Advanced Encryption Standard New Instructions (AES-NI). Para verificar o suporte a AES-NI, execute o seguinte comando:

```
grep "aes" /proc/cpuinfo
```

Se nada for retornado, seu processador não suporta AES-NI. Para mais informações sobre AES-NI, visite: "Intel: instruções do Padrão de Criptografia Avançada (AES-NI)".

Ativar criptografia LUKS

Você pode habilitar a criptografia por volume, do lado do host, usando o Linux Unified Key Setup (LUKS) para volumes ONTAP SAN e ONTAP SAN ECONOMY.

Passos

1. Defina os atributos de criptografia LUKS na configuração do backend. Para obter mais informações sobre as opções de configuração do backend para ONTAP SAN, consulte "Opções de configuração do ONTAP SAN".

```
{
  "storage": [
    {
      "labels": {
        "luks": "true"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "true"
      }
    },
    {
      "labels": {
        "luks": "false"
      },
      "zone": "us_east_1a",
      "defaults": {
        "luksEncryption": "false"
      }
    }
  ]
}
```

2. Use `parameters.selector` para definir os pools de armazenamento usando criptografia LUKS. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
```

3. Crie um segredo que contenha a senha LUKS. Por exemplo:

```
kubectl -n trident create -f luks-pvc1.yaml
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: A
  luks-passphrase: secretA
```

Limitações

Volumes criptografados com LUKS não podem aproveitar a deduplicação e a compressão do ONTAP.

Configuração de backend para importação de volumes LUKS

Para importar um volume LUKS, você deve definir `luksEncryption` para (`true` no backend. A opção `luksEncryption` informa ao Trident se o volume é compatível com LUKS (`true` ou não compatível com LUKS (`false`, conforme mostrado no exemplo a seguir.

```
version: 1
storageDriverName: ontap-san
managementLIF: 10.0.0.1
dataLIF: 10.0.0.2
svm: trident_svm
username: admin
password: password
defaults:
  luksEncryption: 'true'
  spaceAllocation: 'false'
  snapshotPolicy: default
  snapshotReserve: '10'
```

Configuração de PVC para importação de volumes LUKS

Para importar volumes LUKS dinamicamente, defina a anotação `trident.netapp.io/luksEncryption` para `true` e inclua uma classe de armazenamento habilitada para LUKS no PVC, conforme mostrado neste exemplo.

```
kind: PersistentVolumeClaim
apiVersion: v1
metadata:
  name: luks-pvc
  namespace: trident
  annotations:
    trident.netapp.io/luksEncryption: "true"
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi
  storageClassName: luks-sc
```

Rotacionar uma frase secreta LUKS

Você pode alternar a senha LUKS e confirmar a rotação.

> Não se esqueça de uma senha até verificar se ela não está mais sendo referenciada por nenhum volume, snapshot ou segredo. Se uma senha referenciada for perdida, você poderá não conseguir montar o volume e os dados permanecerão criptografados e inacessíveis.

Sobre esta tarefa

A rotação da senha LUKS ocorre quando um pod que monta o volume é criado após a especificação de uma nova senha LUKS. Quando um novo pod é criado, Trident compara a senha LUKS no volume com a senha ativa no segredo.

- Se a senha no volume não corresponder à senha ativa no segredo, a rotação ocorrerá.
- Se a senha no volume corresponder à senha ativa no segredo, o `previous-luks-passphrase` parâmetro será ignorado.

Passos

1. Adicione os `node-publish-secret-name` e `node-publish-secret-namespace` parâmetros StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: csi-san
provisioner: csi.trident.netapp.io
parameters:
  trident.netapp.io/backendType: "ontap-san"
  csi.storage.k8s.io/node-stage-secret-name: luks
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-publish-secret-name: luks
  csi.storage.k8s.io/node-publish-secret-namespace: ${pvc.namespace}
```

2. Identifique as frases secretas existentes no volume ou no snapshot.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["A"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["A"]
```

3. Atualize o segredo LUKS do volume para especificar as senhas nova e anterior. Certifique-se de que `previous-luke-passphrase-name` e `previous-luks-passphrase` correspondam à senha anterior.

```
apiVersion: v1
kind: Secret
metadata:
  name: luks-pvc1
stringData:
  luks-passphrase-name: B
  luks-passphrase: secretB
  previous-luks-passphrase-name: A
  previous-luks-passphrase: secretA
```

4. Crie um novo pod montando o volume. Isso é necessário para iniciar a rotação.
5. Verifique se a senha foi rotacionada.

Volume

```
tridentctl -d get volume luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>

...luksPassphraseNames:["B"]
```

Snapshot

```
tridentctl -d get snapshot luks-pvc1
GET http://127.0.0.1:8000/trident/v1/volume/<volumeID>/<snapshotID>

...luksPassphraseNames:["B"]
```

Resultados

A senha foi rotacionada quando apenas a nova senha foi retornada no volume e no snapshot.

> Se duas senhas forem retornadas, por exemplo `luksPassphraseNames: ["B", "A"]`, a rotação estará incompleta. Você pode acionar um novo pod para tentar concluir a rotação.

Habilitar expansão de volume

Você pode habilitar a expansão de volume em um volume criptografado com LUKS.

Passos

1. Ative o `CSINodeExpandSecret`feature gate (beta 1.25+). Consulte "Kubernetes 1.25: use segredos para expansão de volumes CSI orientada a nós" para obter detalhes.
2. Adicione os `node-expand-secret-name` e `node-expand-secret-namespace` parâmetros StorageClass. Por exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: luks
provisioner: csi.trident.netapp.io
parameters:
  selector: "luks=true"
  csi.storage.k8s.io/node-stage-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-stage-secret-namespace: ${pvc.namespace}
  csi.storage.k8s.io/node-expand-secret-name: luks-${pvc.name}
  csi.storage.k8s.io/node-expand-secret-namespace: ${pvc.namespace}
allowVolumeExpansion: true
```

Resultados

Ao iniciar a expansão de storage online, o kubelet passa as credenciais apropriadas para o driver.

Criptografia em trânsito Kerberos

Usando criptografia em trânsito Kerberos, você pode melhorar a segurança do acesso aos dados habilitando a criptografia para o tráfego entre seu cluster gerenciado e o storage backend.

Trident oferece suporte à criptografia Kerberos para ONTAP como backend de storage:

- **On-premise ONTAP** - Trident oferece suporte à criptografia Kerberos em conexões NFSv3 e NFSv4 de clusters Red Hat OpenShift e Kubernetes upstream para volumes ONTAP locais.

Você pode criar, excluir, redimensionar, criar snapshots, clonar, clonar em modo somente leitura e importar volumes que utilizam criptografia NFS.

Configurar criptografia Kerberos em trânsito com volumes ONTAP locais

Você pode habilitar a criptografia Kerberos no tráfego de storage entre seu cluster gerenciado e um backend de storage ONTAP local.

> A criptografia Kerberos para o tráfego NFS com backends de storage ONTAP locais só é compatível usando o driver de storage `ontap-nas`.

Antes de começar

- Certifique-se de ter acesso ao utilitário `tridentctl`.
- Certifique-se de ter acesso de administrador ao backend de storage ONTAP.
- Certifique-se de saber o nome do(s) volume(s) que você compartilhará do backend de storage ONTAP.
- Certifique-se de ter preparado a máquina virtual de storage ONTAP para suportar a criptografia Kerberos para volumes NFS. Consulte "Habilite o Kerberos em um dataLIF" para obter instruções.
- Certifique-se de que todos os volumes NFSv4 que você usa com criptografia Kerberos estejam configurados corretamente. Consulte a seção Configuração de Domínio NFSv4 da NetApp (página 13) do "NetApp NFSv4 Melhorias e Guia de Práticas Recomendadas".

Adicionar ou modificar políticas de exportação do ONTAP

Você precisa adicionar regras às políticas de exportação ONTAP existentes ou criar novas políticas de exportação que ofereçam suporte à criptografia Kerberos para o volume raiz da máquina virtual de storage ONTAP, bem como para quaisquer volumes ONTAP compartilhados com o cluster Kubernetes upstream. As regras de política de exportação que você adicionar, ou as novas políticas de exportação que você criar, precisam oferecer suporte aos seguintes protocolos de acesso e permissões de acesso:

Protocolos de acesso

Configure a política de exportação com os protocolos de acesso NFS, NFSv3 e NFSv4.

Detalhes de acesso

Você pode configurar uma das três versões diferentes de criptografia Kerberos, dependendo das suas necessidades para o volume:

- **Kerberos 5** - (autenticação e criptografia)
- **Kerberos 5i** - (autenticação e criptografia com proteção de identidade)
- **Kerberos 5p** - (autenticação e criptografia com proteção de identidade e privacidade)

Configure a regra de política de exportação do ONTAP com as permissões de acesso apropriadas. Por exemplo, se os clusters forem montar os volumes NFS com uma combinação de criptografia Kerberos 5i e Kerberos 5p, use as seguintes configurações de acesso:

Tipo	Acesso somente leitura	Acesso de leitura/gravação	Acesso de superusuário
UNIX	Habilitado	Habilitado	Habilitado
Kerberos 5i	Habilitado	Habilitado	Habilitado
Kerberos 5p	Habilitado	Habilitado	Habilitado

Consulte a seguinte documentação para saber como criar políticas de exportação do ONTAP e regras de políticas de exportação:

- "Criar uma política de exportação"
- "Adicionar uma regra a uma política de exportação"

Crie um backend de storage

Você pode criar uma configuração de backend de storage Trident que inclua a capacidade de criptografia Kerberos.

Sobre esta tarefa

Ao criar um arquivo de configuração de storage backend que configura a criptografia Kerberos, você pode especificar uma das três versões diferentes de criptografia Kerberos usando o `spec.nfsMountOptions` parâmetro:

- `spec.nfsMountOptions: sec=krb5` (autenticação e criptografia)
- `spec.nfsMountOptions: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `spec.nfsMountOptions: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Especifique apenas um nível de Kerberos. Se você especificar mais de um nível de criptografia Kerberos na lista de parâmetros, somente a primeira opção será usada.

Passos

1. No cluster gerenciado, crie um arquivo de configuração de backend de storage usando o exemplo a seguir. Substitua os valores entre colchetes <> com informações do seu ambiente:

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-ontap-nas-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>
---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-ontap-nas
spec:
  version: 1
  storageDriverName: "ontap-nas"
  managementLIF: <STORAGE_VM_MGMT_LIF_IP_ADDRESS>
  dataLIF: <PROTOCOL_LIF_FQDN_OR_IP_ADDRESS>
  svm: <STORAGE_VM_NAME>
  username: <STORAGE_VM_USERNAME_CREDENTIAL>
  password: <STORAGE_VM_PASSWORD_CREDENTIAL>
  nasType: nfs
  nfsMountOptions: ["sec=krb5i"] #can be krb5, krb5i, or krb5p
  qtreesPerFlexvol:
  credentials:
    name: backend-ontap-nas-secret
```

2. Use o arquivo de configuração que você criou na etapa anterior para criar o backend:

```
tridentctl create backend -f <backend-configuration-file>
```

Se a criação do backend falhar, há algo errado com a configuração do backend. Você pode visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Crie uma storage class

Você pode criar uma classe de armazenamento para provisionar volumes com criptografia Kerberos.

Sobre esta tarefa

Ao criar um objeto de classe de armazenamento, você pode especificar uma das três versões diferentes de criptografia Kerberos usando o `mountOptions`parâmetro:

- `mountOptions: sec=krb5` (autenticação e criptografia)
- `mountOptions: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `mountOptions: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Especifique apenas um nível Kerberos. Se você especificar mais de um nível de criptografia Kerberos na lista de parâmetros, somente a primeira opção será usada. Se o nível de criptografia que você especificou na configuração do backend de storage for diferente do nível que você especificar no objeto da classe de storage, o objeto da classe de storage terá precedência.

Passos

1. Crie um objeto Kubernetes StorageClass, usando o seguinte exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: ontap-nas-sc
provisioner: csi.trident.netapp.io
mountOptions:
  - sec=krb5i #can be krb5, krb5i, or krb5p
parameters:
  backendType: ontap-nas
  storagePools: ontapnas_pool
  trident.netapp.io/nasType: nfs
allowVolumeExpansion: true
```

2. Crie a classe de armazenamento:

```
kubectl create -f sample-input/storage-class-ontap-nas-sc.yaml
```

3. Certifique-se de que a classe de armazenamento foi criada:

```
kubectl get sc ontap-nas-sc
```

   Você deverá ver uma saída semelhante à seguinte:

```
NAME          PROVISIONER             AGE
ontap-nas-sc    csi.trident.netapp.io   15h
```

Provisionar volumes

Após criar um backend de armazenamento e uma classe de armazenamento, você pode provisionar um volume. Para obter instruções, consulte "Provisionar um volume".

Configurar criptografia Kerberos em trânsito com volumes do Azure NetApp Files

Você pode habilitar a criptografia Kerberos no tráfego de armazenamento entre seu cluster gerenciado e um único back-end de armazenamento Azure NetApp Files ou um pool virtual de back-ends de armazenamento Azure NetApp Files.

Antes de começar

- Certifique-se de ter habilitado Trident no cluster Red Hat OpenShift gerenciado.
- Certifique-se de ter acesso ao utilitário `tridentctl`.
- Certifique-se de ter preparado o backend de armazenamento Azure NetApp Files para criptografia Kerberos, observando os requisitos e seguindo as instruções em "Documentação do Azure NetApp Files".
- Certifique-se de que todos os volumes NFSv4 que você usa com criptografia Kerberos estejam configurados corretamente. Consulte a seção Configuração de Domínio NFSv4 da NetApp (página 13) do "NetApp NFSv4 Melhorias e Guia de Práticas Recomendadas".

Crie um backend de storage

Você pode criar uma configuração de back-end de armazenamento do Azure NetApp Files que inclua a capacidade de criptografia Kerberos.

Sobre esta tarefa

Ao criar um arquivo de configuração de backend de storage que configura a criptografia Kerberos, você pode defini-lo para que seja aplicado em um dos dois níveis possíveis:

- O **nível de backend de armazenamento** usando o `spec.kerberos` campo
- O **nível do pool virtual** usando o `spec.storage.kerberos` campo

Ao definir a configuração no nível do pool virtual, o pool é selecionado usando o rótulo na storage class.

Em qualquer um dos níveis, você pode especificar uma das três versões diferentes de criptografia Kerberos:

- `kerberos: sec=krb5` (autenticação e criptografia)
- `kerberos: sec=krb5i` (autenticação e criptografia com proteção de identidade)
- `kerberos: sec=krb5p` (autenticação e criptografia com proteção de identidade e privacidade)

Passos

1. No cluster gerenciado, crie um arquivo de configuração de backend de storage usando um dos exemplos a seguir, dependendo de onde você precisa definir o backend de storage (nível do backend de storage ou nível do pool virtual). Substitua os valores entre colchetes <> com as informações do seu ambiente:

Exemplo de nível de storage backend

```
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

Exemplo de nível de pool virtual

```
---
apiVersion: v1
kind: Secret
metadata:
  name: backend-tbc-secret
type: Opaque
stringData:
  clientID: <CLIENT_ID>
  clientSecret: <CLIENT_SECRET>

---
apiVersion: trident.netapp.io/v1
kind: TridentBackendConfig
metadata:
  name: backend-tbc
spec:
  version: 1
  storageDriverName: azure-netapp-files
  subscriptionID: <SUBSCRIPTION_ID>
  tenantID: <TENANT_ID>
  location: <AZURE_REGION_LOCATION>
  serviceLevel: Standard
  networkFeatures: Standard
  capacityPools: <CAPACITY_POOL>
  resourceGroups: <RESOURCE_GROUP>
  netappAccounts: <NETAPP_ACCOUNT>
  virtualNetwork: <VIRTUAL_NETWORK>
  subnet: <SUBNET>
  nasType: nfs
  storage:
    - labels:
        type: encryption
      kerberos: sec=krb5i #can be krb5, krb5i, or krb5p
  credentials:
    name: backend-tbc-secret
```

2. Use o arquivo de configuração que você criou na etapa anterior para criar o backend:

```
tridentctl create backend -f <backend-configuration-file>
```

Se a criação do backend falhar, há algo errado com a configuração do backend. Você pode visualizar os logs para determinar a causa executando o seguinte comando:

```
tridentctl logs
```

Após identificar e corrigir o problema com o arquivo de configuração, você pode executar o comando create novamente.

Crie uma storage class

Você pode criar uma classe de armazenamento para provisionar volumes com criptografia Kerberos.

Passos

1. Crie um objeto Kubernetes StorageClass, usando o seguinte exemplo:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: sc-nfs
provisioner: csi.trident.netapp.io
parameters:
  backendType: azure-netapp-files
  trident.netapp.io/nasType: nfs
  selector: type=encryption
```

2. Crie a classe de armazenamento:

```
kubectl create -f sample-input/storage-class-sc-nfs.yaml
```

3. Certifique-se de que a classe de armazenamento foi criada:

```
kubectl get sc -sc-nfs
```

Você deverá ver uma saída semelhante à seguinte:

```
NAME          PROVISIONER             AGE
sc-nfs        csi.trident.netapp.io   15h
```

Provisionar volumes

Após criar um backend de armazenamento e uma classe de armazenamento, você pode provisionar um volume. Para obter instruções, consulte "Provisionar um volume".

Proteja aplicativos com Trident Protect

Saiba mais sobre Trident Protect

NetApp Trident Protect oferece recursos avançados de gerenciamento de dados de aplicativos que aprimoram a funcionalidade e a disponibilidade de aplicativos Kubernetes com estado, com suporte dos sistemas de armazenamento NetApp ONTAP e do provisionador de armazenamento NetApp Trident CSI. Trident Protect simplifica o gerenciamento, a proteção e a movimentação de cargas de trabalho conteinerizadas em nuvens públicas e ambientes locais. Ele também oferece recursos de automação por meio de sua API e CLI.

Você pode proteger aplicativos com Trident Protect criando recursos personalizados (CRs) ou usando a Trident Protect CLI.

Qual é o próximo passo?

Você pode conhecer os requisitos do Trident Protect antes de instalá-lo:

- "Requisitos do Trident Protect"

Instalar Trident Protect

Requisitos do Trident Protect

Comece verificando a prontidão do seu ambiente operacional, clusters de aplicativos, aplicativos e licenças. Certifique-se de que seu ambiente atenda a esses requisitos para implantar e operar Trident Protect.

Compatibilidade do cluster Kubernetes com o Trident Protect

Trident Protect é compatível com uma ampla gama de ofertas de Kubernetes totalmente gerenciadas e autogerenciadas, incluindo:

- Amazon Elastic Kubernetes Service (EKS)
- Google Kubernetes Engine (GKE)
- Microsoft Azure Kubernetes Service (AKS)
- Red Hat OpenShift
- SUSE Rancher
- VMware Tanzu Portfolio
- Kubernetes upstream

- Os backups do Trident Protect são suportados apenas em nós de computação Linux. Nós de computação Windows não são suportados para operações de backup.
- Certifique-se de que o cluster no qual você instala Trident Protect esteja configurado com um controlador de snapshot em execução e os CRDs relacionados. Para instalar um controlador de snapshot, consulte "estas instruções".
- Certifique-se de que exista pelo menos um VolumeSnapshotClass. Para obter mais informações, consulte "VolumeSnapshotClass".

Compatibilidade do backend de armazenamento Trident Protect

Trident Protect é compatível com os seguintes storage backends:

- Amazon FSx for NetApp ONTAP
- Cloud Volumes ONTAP
- Matrizes de armazenamento ONTAP
- Google Cloud NetApp Volumes
- Azure NetApp Files

Certifique-se de que seu storage backend atenda aos seguintes requisitos:

- Certifique-se de que o armazenamento NetApp conectado ao cluster esteja usando Trident 24.02 ou mais recente (Trident 24.10 é recomendado).
- Certifique-se de ter um backend de storage NetApp ONTAP.
- Certifique-se de ter configurado um storage de objetos para armazenar backups.
- Crie todos os namespaces de aplicativos que você planeja usar para aplicativos ou operações de gerenciamento de dados de aplicativos. Trident Protect não cria esses namespaces para você; se você especificar um namespace inexistente em um recurso personalizado, a operação falhará.

Requisitos para volumes nas-economy

Trident Protect oferece suporte a operações de backup e restauração em volumes nas-economy. Snapshots, clones e SnapMirror replicação para volumes nas-economy não são suportados atualmente. Você precisa habilitar um diretório de snapshot para cada volume nas-economy que planeja usar com Trident Protect.

Algumas aplicações não são compatíveis com volumes que utilizam um diretório de snapshot. Para essas aplicações, você precisa ocultar o diretório de snapshot executando o seguinte comando no sistema de storage ONTAP:

```
nfs modify -vserver <svm> -v3-hide-snapshot enabled
```

Você pode habilitar o diretório de snapshots executando o seguinte comando para cada volume nas-economy, substituindo `<volume-UUID>` pelo UUID do volume que deseja alterar:

```
tridentctl update volume <volume-UUID> --snapshot-dir=true --pool-level
=true -n trident
```

Você pode habilitar diretórios de snapshots por padrão para novos volumes definindo a opção de configuração do backend Trident `snapshotDir` para `true`. Volumes existentes não são afetados.

Protegendo dados com KubeVirt VMs

Trident Protect oferece recursos de congelamento e descongelamento do sistema de arquivos para máquinas virtuais KubeVirt durante operações de proteção de dados para garantir a consistência de dados. O método de configuração e o comportamento padrão para operações de congelamento de máquinas virtuais variam entre as versões do Trident Protect, sendo que as versões mais recentes oferecem configuração simplificada por meio de parâmetros do Helm chart.

Durante as operações de restauração, quaisquer `VirtualMachineSnapshots` criados para uma máquina virtual (VM) não são restaurados.

Trident Protect 25.10 e mais recentes

Trident Protect congela e descongela automaticamente os sistemas de arquivos KubeVirt durante as operações de proteção de dados para garantir a consistência. A partir do Trident Protect 25.10, você pode desativar esse comportamento usando o parâmetro `vm.freeze` durante a instalação do Helm chart. O parâmetro está ativado por padrão.

```
helm install ... --set vm.freeze=false ...
```

Trident Protect 24.10.1 a 25.06

A partir do Trident Protect 24.10.1, o Trident Protect congela e descongela automaticamente os sistemas de arquivos KubeVirt durante as operações de proteção de dados. Opcionalmente, você pode desativar esse comportamento automático usando o seguinte comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=false -n trident-protect
```

Trident Protect 24.10

Trident Protect 24.10 não garante automaticamente um estado consistente para os sistemas de arquivos de KubeVirt VM durante as operações de proteção de dados. Se você deseja proteger os dados da sua KubeVirt VM usando Trident Protect 24.10, precisa habilitar manualmente a funcionalidade de congelamento/descongelamento dos sistemas de arquivos antes da operação de proteção de dados. Isso garante que os sistemas de arquivos estejam em um estado consistente.

Você pode configurar Trident Protect 24.10 para gerenciar o congelamento e descongelamento do sistema de arquivos da VM durante as operações de proteção de dados "configurando virtualização" e, em seguida, usar o seguinte comando:

```
kubectl set env deployment/trident-protect-controller-manager
NEPTUNE_VM_FREEZE=true -n trident-protect
```

Requisitos para SnapMirror replication

NetApp SnapMirror a replicação está disponível para uso com Trident Protect para as seguintes soluções ONTAP:

- Sistemas NetApp FAS, AFF e ASA locais. A replicação SnapMirror com Trident protect não é atualmente suportada para sistemas ASA r2.
- NetApp ONTAP Select
- NetApp Cloud Volumes ONTAP
- Amazon FSx for NetApp ONTAP

Requisitos do cluster ONTAP para replicação SnapMirror

Certifique-se de que seu cluster ONTAP atenda aos seguintes requisitos caso planeje usar a replicação SnapMirror:

- **NetApp Trident**: NetApp Trident deve existir em ambos os clusters Kubernetes de origem e destino que utilizam ONTAP como backend. Trident Protect oferece suporte à replicação com a tecnologia NetApp SnapMirror usando classes de armazenamento suportadas pelos seguintes drivers:
  - `ontap-nas`: NFS
  - `ontap-san`: iSCSI
  - `ontap-san`: FC
  - `ontap-san`: NVMe/TCP (requer versão mínima do ONTAP 9.15.1)
- **Licenças**: As licenças assíncronas do ONTAP SnapMirror usando o pacote Data Protection devem estar habilitadas tanto nos clusters ONTAP de origem quanto de destino. Consulte "SnapMirror visão geral do licenciamento no ONTAP" para mais informações.

  A partir do ONTAP 9.10.1, todas as licenças são fornecidas como um arquivo de licença NetApp (NLF), que é um único arquivo que habilita vários recursos. Consulte "Licenças incluídas com o ONTAP One" para mais informações.

  Apenas a proteção assíncrona SnapMirror é suportada.

Considerações de peering para replicação SnapMirror

Certifique-se de que seu ambiente atenda aos seguintes requisitos caso planeje usar o peering de storage backend:

- **Cluster e SVM**: Os backends de armazenamento ONTAP devem estar emparelhados. Consulte "Visão geral do peering de cluster e SVM" para mais informações.

  Certifique-se de que os nomes SVM usados na relação de replicação entre dois clusters ONTAP sejam únicos.

- **NetApp Trident e SVM**: Os SVMs remotos emparelhados devem estar disponíveis para NetApp Trident no cluster de destino.
- **Backends gerenciados**: Você precisa adicionar e gerenciar backends de armazenamento ONTAP no Trident Protect para criar uma relação de replicação.

Configuração do Trident / ONTAP para replicação SnapMirror

Trident Protect exige que você configure pelo menos um storage backend que suporte replicação para ambos os clusters de origem e destino. Se os clusters de origem e destino forem os mesmos, o aplicativo de destino deve usar um storage backend diferente do aplicativo de origem para a melhor resiliência.

Requisitos do cluster Kubernetes para SnapMirror replication

Certifique-se de que seus clusters Kubernetes atendam aos seguintes requisitos:

- **AppVault acessibilidade**: Tanto o cluster de origem quanto o cluster de destino devem ter acesso à rede para ler e gravar na AppVault para replicação de objetos da aplicação.
- **Conectividade de rede**: Configure regras de firewall, permissões de bucket e listas de permissão de IP para habilitar a comunicação entre ambos os clusters e o AppVault através das WANs.

  Muitos ambientes corporativos implementam políticas de firewall rigorosas em conexões WAN. Verifique esses requisitos de rede com sua equipe de infraestrutura antes de configurar a replicação.

Instalar e configurar Trident Protect

Se o seu ambiente atender aos requisitos do Trident Protect, você pode seguir estas etapas para instalar o Trident Protect no seu cluster. Você pode obter o Trident Protect da NetApp ou instalá-lo a partir do seu próprio registro privado. Instalar a partir de um registro privado é útil se o seu cluster não puder acessar a Internet.

Instalar Trident Protect

Instale Trident Protect a partir de NetApp

Passos

1. Adicione o repositório Trident:

```
helm repo add netapp-trident-protect
https://netapp.github.io/trident-protect-helm-chart
```

2. Use o Helm para instalar Trident Protect. Substitua `<name-of-cluster>` pelo nome do cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect
```

3. Opcionalmente, para ativar o registro de depuração (recomendado para solução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect
```

   O registro de depuração ajuda NetApp support a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Instale Trident Protect a partir de um registro privado

Você pode instalar Trident Protect a partir de um registro de imagens privado caso seu cluster Kubernetes não tenha acesso à Internet. Nestes exemplos, substitua os valores entre colchetes pelas informações do seu ambiente:

Passos

1. Baixe as seguintes imagens para sua máquina local, atualize as tags e, em seguida, envie-as para seu registro privado:

```
docker.io/netapp/controller:25.10.0
docker.io/netapp/restic:25.10.0
docker.io/netapp/kopia:25.10.0
docker.io/netapp/kopiablockrestore:25.10.0
docker.io/netapp/trident-autosupport:25.10.0
docker.io/netapp/exechook:25.10.0
docker.io/netapp/resourcebackup:25.10.0
docker.io/netapp/resourcerestore:25.10.0
docker.io/netapp/resourcedelete:25.10.0
docker.io/netapp/trident-protect-utils:v1.0.0
```

Por exemplo:

```
docker pull docker.io/netapp/controller:25.10.0
```

```
docker tag docker.io/netapp/controller:25.10.0 <private-registry-url>/controller:25.10.0
```

```
docker push <private-registry-url>/controller:25.10.0
```

> Para obter o Helm chart, primeiro baixe o Helm chart em uma máquina com acesso à internet usando `helm pull trident-protect --version 100.2510.0 --repo https://netapp.github.io/trident-protect-helm-chart`, depois copie o arquivo resultante `trident-protect-100.2510.0.tgz` para seu ambiente offline e instale usando `helm install trident-protect ./trident-protect-100.2510.0.tgz` em vez da referência ao repositório na etapa final.

2. Crie o namespace do sistema Trident Protect:

```
kubectl create ns trident-protect
```

3. Faça login no registro:

```
helm registry login <private-registry-url> -u <account-id> -p <api-token>
```

4. Crie um segredo de pull para usar na autenticação do registro privado:

```
kubectl create secret docker-registry regcred --docker-username=<registry-username> --docker-password=<api-token> -n trident-protect --docker-server=<private-registry-url>
```

5. Adicione o repositório Trident:

```
helm repo add netapp-trident-protect https://netapp.github.io/trident-protect-helm-chart
```

6. Crie um arquivo chamado `protectValues.yaml`. Certifique-se de que ele contenha as seguintes configurações do Trident Protect:

```
---
imageRegistry: <private-registry-url>
imagePullSecrets:
  - name: regcred
```

> Os `imageRegistry` e `imagePullSecrets` valores se aplicam a todas as imagens de componentes, incluindo `resourcebackup` e `resourcerestore`. Se você enviar imagens para um caminho de repositório específico em seu registro (por exemplo, `example.com:443/my-repo`), inclua o caminho completo no campo de registro. Isso garantirá que todas as imagens sejam obtidas de `<private-registry-url>/<image-name>:<tag>`.

7. Use o Helm para instalar Trident Protect. Substitua `<name_of_cluster>` pelo nome do cluster, que será atribuído ao cluster e usado para identificar os backups e snapshots do cluster:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name_of_cluster> --version 100.2510.0 --create
-namespace --namespace trident-protect -f protectValues.yaml
```

8. Opcionalmente, para ativar o registro de depuração (recomendado para solução de problemas), use:

```
helm install trident-protect netapp-trident-protect/trident-protect
--set clusterName=<name-of-cluster> --set logLevel=debug --version
100.2510.0 --create-namespace --namespace trident-protect -f
protectValues.yaml
```

   O registro de depuração ajuda NetApp support a solucionar problemas sem exigir alterações no nível de registro ou reprodução do problema.

> Para opções adicionais de configuração do gráfico Helm, incluindo configurações do AutoSupport e filtragem de namespace, consulte "Personalizar a instalação do Trident Protect".

Instale o plugin Trident Protect CLI

Você pode usar o plugin de linha de comando Trident Protect, que é uma extensão do utilitário Trident `tridentctl`, para criar e interagir com recursos personalizados (CRs) do Trident Protect.

Instale o plugin Trident Protect CLI

Antes de usar o utilitário de linha de comando, você precisa instalá-lo na máquina que usa para acessar seu cluster. Siga estas etapas, dependendo se sua máquina usa uma CPU x64 ou ARM.

Baixar plugin para CPUs Linux AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-amd64
```

Baixar plugin para CPUs Linux ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-linux-arm64
```

Baixar plugin para CPUs Mac AMD64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-amd64
```

Baixar plugin para CPUs Mac ARM64

Passos

1. Baixe o plugin Trident Protect CLI:

```
curl -L -o tridentctl-protect https://github.com/NetApp/tridentctl-protect/releases/download/25.10.0/tridentctl-protect-macos-arm64
```

1. Habilite as permissões de execução para o binário do plugin:

```
chmod +x tridentctl-protect
```

2. Copie o binário do plugin para um local definido na sua variável PATH. Por exemplo, `/usr/bin` ou `/usr/local/bin` (você pode precisar de privilégios elevados):

```
cp ./tridentctl-protect /usr/local/bin/
```

3. Opcionalmente, você pode copiar o binário do plugin para um local no seu diretório base. Nesse caso, é recomendável garantir que o local faça parte da sua variável PATH:

```
cp ./tridentctl-protect ~/bin/
```

Copiar o plugin para um local na sua variável PATH permite que você utilize o plugin digitando `tridentctl-protect` ou `tridentctl protect` de qualquer local.

Veja a ajuda do plugin Trident CLI

Você pode usar os recursos de ajuda incorporado do plugin para obter ajuda detalhada sobre as funcionalidades do plugin:

Passos

1. Use a função de ajuda para visualizar as instruções de utilização:

```
tridentctl-protect help
```

Ativar o preenchimento automático de comandos

Após instalar o plugin Trident Protect CLI, você pode ativar o recurso de autocompletar para determinados comandos.

Ative o recurso de autocompletar para o shell Bash

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion bash > tridentctl-completion.bash
```

2. Crie um novo diretório em seu diretório base para conter o script:

```
mkdir -p ~/.bash/completions
```

3. Mova o script baixado para o diretório `~/.bash/completions`:

```
mv tridentctl-completion.bash ~/.bash/completions/
```

4. Adicione a seguinte linha ao arquivo `~/.bashrc` no seu diretório base:

```
source ~/.bash/completions/tridentctl-completion.bash
```

Ative o recurso de autocompletar para o Z shell

Passos

1. Crie o script de conclusão:

```
tridentctl-protect completion zsh > tridentctl-completion.zsh
```

2. Crie um novo diretório em seu diretório base para conter o script:

```
mkdir -p ~/.zsh/completions
```

3. Mova o script baixado para o diretório `~/.zsh/completions`:

```
mv tridentctl-completion.zsh ~/.zsh/completions/
```

4. Adicione a seguinte linha ao arquivo `~/.zprofile` no seu diretório base:

```
source ~/.zsh/completions/tridentctl-completion.zsh
```

Resultado

Na sua próxima sessão de login no shell, você pode usar o recurso de autocompletar comandos com o plugin tridentctl-protect.

Personalizar a instalação do Trident Protect

Você pode personalizar a configuração padrão do Trident Protect para atender aos requisitos específicos do seu ambiente.

Especifique os limites de recursos do contêiner Trident Protect

Você pode usar um arquivo de configuração para especificar limites de recursos para os contêineres do Trident Protect após instalar o Trident Protect. Definir limites de recursos permite controlar quanto dos recursos do cluster são consumidos pelas operações do Trident Protect.

Passos

1. Crie um arquivo chamado `resourceLimits.yaml`.
2. Preencha o arquivo com as opções de limite de recursos para os contêineres do Trident Protect de acordo com as necessidades do seu ambiente.

   O seguinte arquivo de configuração mostra as configurações disponíveis e contém os valores padrão para cada limite de recurso:

```
---
jobResources:
  defaults:
    limits:
      cpu: 8000m
      memory: 10000Mi
      ephemeralStorage: ""
    requests:
      cpu: 100m
      memory: 100Mi
      ephemeralStorage: ""
  resticVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  resticVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

```
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeBackup:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
  kopiaVolumeRestore:
    limits:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
    requests:
      cpu: ""
      memory: ""
      ephemeralStorage: ""
```

3. Aplique os valores do arquivo `resourceLimits.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f resourceLimits.yaml --reuse-values
```

Personalizar restrições de contexto de segurança

Você pode usar um arquivo de configuração para modificar as restrições de contexto de segurança (OpenShift SCCs) para os contêineres do Trident Protect após instalar o Trident Protect. Essas restrições definem limitações de segurança para pods em um cluster Red Hat OpenShift.

Passos

1. Crie um arquivo chamado `sccconfig.yaml`.
2. Adicione a opção SCC ao arquivo e modifique os parâmetros de acordo com as necessidades do seu ambiente.

   O exemplo a seguir mostra os valores padrão dos parâmetros para a opção SCC:

```
scc:
  create: true
  name: trident-protect-job
  priority: 1
```

Esta tabela descreve os parâmetros para a opção SCC:

Parâmetro	Descrição	Padrão
criar	Determina se um recurso SCC pode ser criado. Um recurso SCC será criado somente se `scc.create` estiver definido como `true` e o processo de instalação do Helm identificar um ambiente OpenShift. Se não estiver operando em OpenShift, ou se `scc.create` estiver definido como `false`, nenhum recurso SCC será criado.	verdadeiro
nome	Especifica o nome do SCC.	trident-protect-job
prioridade	Define a prioridade do SCC. SCCs com valores de prioridade mais altos são avaliados antes daqueles com valores mais baixos.	1

3. Aplique os valores do arquivo `sccconfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f sccconfig.yaml --reuse-values
```

Isso substituirá os valores padrão pelos valores especificados no arquivo `sccconfig.yaml`.

Configurar configurações adicionais do helm chart do Trident Protect

Você pode personalizar as configurações do AutoSupport e o filtro de namespace para atender às suas necessidades específicas. A tabela a seguir descreve os parâmetros de configuração disponíveis:

Parâmetro	Tipo	Descrição
AutoSupport.proxy	string	Configura uma URL de proxy para conexões do NetApp AutoSupport. Use isto para rotear uploads de pacotes de suporte por meio de um servidor proxy. Exemplo: `http://my.proxy.url`.

Parâmetro	Tipo	Descrição
AutoSupport.insecure	booleano	Ignora a verificação TLS para conexões proxy do AutoSupport quando definido como `true`. Use somente para conexões proxy inseguras. (padrão: `false`)
AutoSupport.enabled	booleano	Ativa ou desativa o envio diário de pacotes do Trident Protect AutoSupport. Quando definido como `false`, os envios diários agendados são desativados, mas você ainda pode gerar pacotes de suporte manualmente. (padrão: `true`)
restoreSkipNamespaceAnnotations	string	Lista de anotações de namespace separadas por vírgulas para excluir das operações de backup e restauração. Permite filtrar namespaces com base em anotações.
restoreSkipNamespaceLabels	string	Lista de rótulos de namespace separados por vírgula para excluir das operações de backup e restauração. Permite filtrar namespaces com base nos rótulos.

Você pode configurar essas opções usando um arquivo de configuração YAML ou parâmetros de linha de comando:

Use o arquivo YAML

Passos

1. Crie um arquivo de configuração e nomeie-o `values.yaml`.
2. No arquivo que você criou, adicione as opções de configuração que deseja personalizar.

```
autoSupport:
  enabled: false
  proxy: http://my.proxy.url
  insecure: true
restoreSkipNamespaceAnnotations: "annotation1,annotation2"
restoreSkipNamespaceLabels: "label1,label2"
```

3. Após preencher o `values.yaml` file com os valores corretos, aplique o arquivo de configuração:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f values.yaml --reuse-values
```

Use a flag da CLI

Passos

1. Utilize o seguinte comando com a flag `--set` para especificar parâmetros individuais:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set autoSupport.enabled=false \
  --set autoSupport.proxy=http://my.proxy.url \
  --set-string
restoreSkipNamespaceAnnotations="{annotation1,annotation2}" \
  --set-string restoreSkipNamespaceLabels="{label1,label2}" \
  --reuse-values
```

Restringir os pods do Trident Protect a nós específicos

Você pode usar a restrição de seleção de nós nodeSelector do Kubernetes para controlar quais dos seus nós estão qualificados para executar pods do Trident Protect, com base nos rótulos dos nós. Por padrão, Trident Protect é restrito a nós que executam Linux. Você pode personalizar ainda mais essas restrições de acordo com suas necessidades.

Passos

1. Crie um arquivo chamado `nodeSelectorConfig.yaml`.
2. Adicione a opção nodeSelector ao arquivo e modifique o arquivo para adicionar ou alterar rótulos de nós para restringir de acordo com as necessidades do seu ambiente. Por exemplo, o seguinte arquivo contém

a restrição padrão do sistema operacional, mas também direciona para uma região e um nome de aplicativo específicos:

```
nodeSelector:
  kubernetes.io/os: linux
  region: us-west
  app.kubernetes.io/name: mysql
```

3. Aplique os valores do arquivo `nodeSelectorConfig.yaml`:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect -f nodeSelectorConfig.yaml --reuse-values
```

Isso substitui as restrições padrão pelas que você especificou no arquivo `nodeSelectorConfig.yaml`.

Gerenciar Trident Protect

Gerencie a autorização e o controle de acesso do Trident

Trident Protect utiliza o modelo Kubernetes de controle de acesso baseado em funções (RBAC). Por padrão, Trident Protect fornece um único namespace de sistema e sua respectiva conta de serviço padrão. Se sua organização possui muitos usuários ou necessidades de segurança específicas, você pode utilizar os recursos de RBAC do Trident Protect para obter um controle mais granular sobre o acesso a recursos e namespaces.

O administrador do cluster sempre tem acesso aos recursos no namespace padrão `trident-protect` e também pode acessar recursos em todos os outros namespaces. Para controlar o acesso a recursos e aplicativos, você precisa criar namespaces adicionais e adicionar recursos e aplicativos a esses namespaces.

Observe que nenhum usuário pode criar CRs de gerenciamento de dados de aplicativos no namespace `trident-protect`padrão. Você precisa criar CRs de gerenciamento de dados de aplicativos em um namespace de aplicativo (como prática recomendada, crie CRs de gerenciamento de dados de aplicativos no mesmo namespace do aplicativo associado).

Somente os administradores devem ter acesso aos objetos de recursos personalizados privilegiados do Trident Protect, que incluem:

- **AppVault**: requer dados de credencial do bucket
- **AutoSupportBundle**: coleta métricas, registros e outros dados confidenciais do Trident Protect
- **AutoSupportBundleSchedule**: gerencia os agendamentos de coleta de logs

Como prática recomendada, use RBAC para restringir o acesso a objetos privilegiados a administradores.

Para obter mais informações sobre como o RBAC regula o acesso a recursos e namespaces, consulte o "Documentação do Kubernetes RBAC".

Para obter informações sobre contas de serviço, consulte o "Documentação da conta de serviço do Kubernetes".

Exemplo: gerenciar o acesso para dois grupos de usuários

Por exemplo, uma organização possui um administrador de cluster, um grupo de usuários de engenharia e um grupo de usuários de marketing. O administrador de cluster executaria as seguintes tarefas para criar um ambiente onde o grupo de engenharia e o grupo de marketing tenham acesso apenas aos recursos atribuídos aos seus respectivos namespaces.

Etapa 1: crie um espaço de nomes para conter os recursos de cada grupo

Criar um namespace permite separar recursos logicamente e controlar melhor quem tem acesso a esses recursos.

Passos

1. Crie um namespace para o grupo de engenharia:

```
kubectl create ns engineering-ns
```

2. Crie um namespace para o grupo de marketing:

```
kubectl create ns marketing-ns
```

Etapa 2: crie novas contas de serviço para interagir com os recursos em cada namespace

Cada novo namespace que você cria vem com uma conta de serviço padrão, mas você deve criar uma conta de serviço para cada grupo de usuários para que possa dividir ainda mais os privilégios entre grupos no futuro, se necessário.

Passos

1. Crie uma conta de serviço para o grupo de engenharia:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: eng-user
  namespace: engineering-ns
```

2. Crie uma conta de serviço para o grupo de marketing:

```
apiVersion: v1
kind: ServiceAccount
metadata:
  name: mkt-user
  namespace: marketing-ns
```

Etapa 3: crie um segredo para cada nova conta de serviço

Uma chave secreta da conta de serviço é usada para autenticar com a conta de serviço e pode ser facilmente excluída e recriada se for comprometida.

Passos

1. Crie um segredo para a conta de serviço de engenharia:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: eng-user
  name: eng-user-secret
  namespace: engineering-ns
type: kubernetes.io/service-account-token
```

2. Crie um segredo para a conta do serviço de marketing:

```
apiVersion: v1
kind: Secret
metadata:
  annotations:
    kubernetes.io/service-account.name: mkt-user
  name: mkt-user-secret
  namespace: marketing-ns
type: kubernetes.io/service-account-token
```

Etapa 4: Crie um objeto RoleBinding para vincular o objeto ClusterRole a cada nova conta de serviço

Um objeto ClusterRole padrão é criado quando você instala Trident Protect. Você pode vincular esse ClusterRole à conta de serviço criando e aplicando um objeto RoleBinding.

Passos

1. Vincule o ClusterRole à conta de serviço de engenharia:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: engineering-ns-tenant-rolebinding
  namespace: engineering-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

2. Vincule o ClusterRole à conta de serviço de marketing:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: marketing-ns-tenant-rolebinding
  namespace: marketing-ns
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: ClusterRole
  name: trident-protect-tenant-cluster-role
subjects:
- kind: ServiceAccount
  name: mkt-user
  namespace: marketing-ns
```

Etapa 5: testar permissões

Teste se as permissões estão corretas.

Passos

1. Confirme se os usuários de engenharia podem acessar os recursos de engenharia:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n engineering-ns
```

2. Confirme que os usuários de engenharia não podem acessar os recursos de marketing:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get applications.protect.trident.netapp.io -n marketing-ns
```

Etapa 6: conceder acesso a objetos AppVault

Para executar tarefas de gerenciamento de dados, como backups e snapshots, o administrador do cluster precisa conceder acesso a objetos AppVault a usuários individuais.

Passos

1. Crie e aplique um arquivo YAML de combinação de AppVault e segredo que conceda a um usuário acesso a um AppVault. Por exemplo, o seguinte CR concede acesso a um AppVault ao usuário `eng-user`:

```
apiVersion: v1
data:
  accessKeyID: <ID_value>
  secretAccessKey: <key_value>
kind: Secret
metadata:
  name: appvault-for-eng-user-only-secret
  namespace: trident-protect
type: Opaque
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: appvault-for-eng-user-only
  namespace: trident-protect # Trident Protect system namespace
spec:
  providerConfig:
    azure:
      accountName: ""
      bucketName: ""
      endpoint: ""
    gcp:
      bucketName: ""
      projectID: ""
    s3:
      bucketName: testbucket
      endpoint: 192.168.0.1:30000
      secure: "false"
      skipCertValidation: "true"
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: appvault-for-eng-user-only-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: appvault-for-eng-user-only-secret
  providerType: GenericS3
```

2. Crie e aplique uma Role CR para permitir que os administradores do cluster concedam acesso a recursos específicos em um namespace. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: Role
metadata:
  name: eng-user-appvault-reader
  namespace: trident-protect
rules:
- apiGroups:
  - protect.trident.netapp.io
  resourceNames:
  - appvault-for-enguser-only
  resources:
  - appvaults
  verbs:
  - get
```

3. Crie e aplique uma RoleBinding CR para vincular as permissões ao usuário eng-user. Por exemplo:

```
apiVersion: rbac.authorization.k8s.io/v1
kind: RoleBinding
metadata:
  name: eng-user-read-appvault-binding
  namespace: trident-protect
roleRef:
  apiGroup: rbac.authorization.k8s.io
  kind: Role
  name: eng-user-appvault-reader
subjects:
- kind: ServiceAccount
  name: eng-user
  namespace: engineering-ns
```

4. Verifique se as permissões estão corretas.

   a. Tentativa de recuperar informações do objeto AppVault para todos os namespaces:

   ```
   kubectl get appvaults -n trident-protect
   --as=system:serviceaccount:engineering-ns:eng-user
   ```

   Você deverá ver uma saída semelhante à seguinte:

```
Error from server (Forbidden): appvaults.protect.trident.netapp.io is
forbidden: User "system:serviceaccount:engineering-ns:eng-user"
cannot list resource "appvaults" in API group
"protect.trident.netapp.io" in the namespace "trident-protect"
```

b. Teste para verificar se o usuário consegue obter as AppVault informações às quais agora tem permissão de acesso:

```
kubectl auth can-i --as=system:serviceaccount:engineering-ns:eng-user
get appvaults.protect.trident.netapp.io/appvault-for-eng-user-only -n
trident-protect
```

Você deverá ver uma saída semelhante à seguinte:

```
yes
```

Resultado

Os usuários aos quais você concedeu permissões do AppVault devem poder usar objetos AppVault autorizados para operações de gerenciamento de dados do aplicativo e não devem poder acessar recursos fora dos namespaces atribuídos nem criar novos recursos aos quais não tenham acesso.

Monitorar recursos do Trident Protect

Você pode usar as ferramentas de código aberto kube-state-metrics, Prometheus e Alertmanager para monitorar a integridade dos recursos protegidos pelo Trident Protect.

O serviço kube-state-metrics gera métricas a partir da comunicação da API do Kubernetes. Usá-lo com Trident Protect expõe informações úteis sobre o estado dos recursos em seu ambiente.

Prometheus é um conjunto de ferramentas que pode ingerir os dados gerados pelo kube-state-metrics e apresentá-los como informações facilmente legíveis sobre esses objetos. Juntos, kube-state-metrics e Prometheus fornecem uma maneira de monitorar a integridade e o status dos recursos que você gerencia com Trident Protect.

Alertmanager é um serviço que recebe os alertas enviados por ferramentas como Prometheus e os encaminha para destinos que você configurar.

> As configurações e orientações incluídas nestas etapas são apenas exemplos; você precisa personalizá-las para corresponder ao seu ambiente. Consulte a seguinte documentação oficial para instruções específicas e suporte:
>
> - "documentação do kube-state-metrics"
> - "Documentação do Prometheus"
> - "Documentação do Alertmanager"

Passo 1: instale as ferramentas de monitoramento

Para habilitar o monitoramento de recursos no Trident Protect, você precisa instalar e configurar kube-state-metrics, Prometheus e Alertmanager.

Instalar kube-state-metrics

Você pode instalar kube-state-metrics usando Helm.

Passos

1. Adicione o gráfico Helm kube-state-metrics. Por exemplo:

```
helm repo add prometheus-community https://prometheus-
community.github.io/helm-charts
helm repo update
```

2. Aplique o CRD ServiceMonitor do Prometheus ao cluster:

```
kubectl apply -f https://raw.githubusercontent.com/prometheus-
operator/prometheus-operator/main/example/prometheus-operator-
crd/monitoring.coreos.com_servicemonitors.yaml
```

3. Crie um arquivo de configuração para o Helm chart (por exemplo, `metrics-config.yaml`). Você pode personalizar a seguinte configuração de exemplo para corresponder ao seu ambiente:

metrics-config.yaml: configuração do Helm chart kube-state-metrics

```
---
extraArgs:
  # Collect only custom metrics
  - --custom-resource-state-only=true

customResourceState:
  enabled: true
  config:
    kind: CustomResourceStateMetrics
    spec:
      resources:
      - groupVersionKind:
          group: protect.trident.netapp.io
          kind: "Backup"
          version: "v1"
        labelsFromPath:
          backup_uid: [metadata, uid]
          backup_name: [metadata, name]
          creation_time: [metadata, creationTimestamp]
        metrics:
        - name: backup_info
          help: "Exposes details about the Backup state"
          each:
            type: Info
            info:
              labelsFromPath:
                appVaultReference: ["spec", "appVaultRef"]
                appReference: ["spec", "applicationRef"]
rbac:
  extraRules:
  - apiGroups: ["protect.trident.netapp.io"]
    resources: ["backups"]
    verbs: ["list", "watch"]

# Collect metrics from all namespaces
namespaces: ""

# Ensure that the metrics are collected by Prometheus
prometheus:
  monitor:
    enabled: true
```

4. Instale o kube-state-metrics implantando o gráfico Helm. Por exemplo:

```
helm install custom-resource -f metrics-config.yaml prometheus-
community/kube-state-metrics --version 5.21.0
```

5. Configure o kube-state-metrics para gerar métricas para os recursos personalizados usados pelo Trident Protect seguindo as instruções em "Documentação do recurso personalizado kube-state-metrics".

Instale o Prometheus

Você pode instalar o Prometheus seguindo as instruções no "Documentação do Prometheus".

Instale o Alertmanager

Você pode instalar o Alertmanager seguindo as instruções no "Documentação do Alertmanager".

Etapa 2: configure as ferramentas de monitoramento para funcionarem em conjunto

Após instalar as ferramentas de monitoramento, você precisa configurá-las para que funcionem juntas.

Passos

1. Integre o kube-state-metrics com o Prometheus. Edite o arquivo de configuração do Prometheus (`prometheus.yaml` e adicione as informações do serviço kube-state-metrics. Por exemplo:

   prometheus.yaml: integração do serviço kube-state-metrics com Prometheus

```
---
apiVersion: v1
kind: ConfigMap
metadata:
  name: prometheus-config
  namespace: trident-protect
data:
  prometheus.yaml: |
    global:
      scrape_interval: 15s
    scrape_configs:
      - job_name: 'kube-state-metrics'
        static_configs:
          - targets: ['kube-state-metrics.trident-protect.svc:8080']
```

2. Configure o Prometheus para encaminhar alertas para o Alertmanager. Edite o arquivo de configuração do Prometheus (`prometheus.yaml` e adicione a seguinte seção:

prometheus.yaml: enviar alertas para o Alertmanager

```
alerting:
  alertmanagers:
    - static_configs:
        - targets:
            - alertmanager.trident-protect.svc:9093
```

Resultado

Prometheus agora pode coletar métricas do kube-state-metrics e enviar alertas para Alertmanager. Agora você está pronto para configurar quais condições acionam um alerta e para onde os alertas devem ser enviados.

Etapa 3: configurar alertas e destinos de alerta

Depois de configurar as ferramentas para funcionarem em conjunto, você precisa configurar que tipo de informação aciona alertas e para onde os alertas devem ser enviados.

Exemplo de alerta: falha no backup

O exemplo a seguir define um alerta crítico que é acionado quando o status do recurso personalizado de backup é definido como `Error` por 5 segundos ou mais. Você pode personalizar este exemplo para corresponder ao seu ambiente e incluir este trecho de YAML em seu arquivo de configuração `prometheus.yaml`:

rules.yaml: defina um alerta do Prometheus para backups com falha

```
rules.yaml: |
  groups:
    - name: fail-backup
        rules:
          - alert: BackupFailed
            expr: kube_customresource_backup_info{status="Error"}
            for: 5s
            labels:
              severity: critical
            annotations:
              summary: "Backup failed"
              description: "A backup has failed."
```

Configure o Alertmanager para enviar alertas para outros canais

Você pode configurar o Alertmanager para enviar notificações para outros canais, como e-mail, PagerDuty, Microsoft Teams ou outros serviços de notificação, especificando a respectiva configuração no arquivo `alertmanager.yaml`.

O exemplo a seguir configura o Alertmanager para enviar notificações a um canal do Slack. Para personalizar este exemplo para o seu ambiente, substitua o valor da `api_url` chave pela URL do webhook do Slack usada em seu ambiente:

alertmanager.yaml: enviar alertas para um canal do Slack

```
data:
  alertmanager.yaml: |
    global:
      resolve_timeout: 5m
    route:
      receiver: 'slack-notifications'
    receivers:
      - name: 'slack-notifications'
        slack_configs:
          - api_url: '<your-slack-webhook-url>'
            channel: '#failed-backups-channel'
            send_resolved: false
```

Gerar um pacote de suporte Trident Protect

Trident Protect permite que os administradores gerem pacotes que incluem informações úteis para o NetApp Support, incluindo logs, métricas e informações de topologia sobre os clusters e aplicativos sob gerenciamento. Se você estiver conectado à internet, poderá enviar pacotes de suporte para o site de suporte da NetApp (NSS) usando um arquivo de recurso personalizado (CR).

Crie um pacote de suporte usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-support-bundle.yaml`).
2. Configurar os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.triggerType**: (*Obrigatório*) Determina se o pacote de suporte é gerado imediatamente ou agendado. A geração agendada do pacote ocorre às 00h UTC. Valores possíveis:
     - Agendado
     - Manual
   - **spec.uploadEnabled**: (*Opcional*) Controla se o pacote de suporte deve ser carregado no site de suporte da NetApp após ser gerado. Se não for especificado, o padrão é `false`. Valores possíveis:
     - verdadeiro
     - falso (default)
   - **spec.dataWindowStart**: (*Opcional*) Uma string de data no formato RFC 3339 que especifica a data e hora em que a janela de dados incluída no pacote de suporte deve começar. Se não for especificado, o padrão é 24 horas atrás. A data mais antiga da janela que você pode especificar é 7 dias atrás.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AutoSupportBundle
metadata:
  name: trident-protect-support-bundle
spec:
  triggerType: Manual
  uploadEnabled: true
  dataWindowStart: 2024-05-05T12:30:00Z
```

3. Após preencher o arquivo `trident-protect-support-bundle.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-support-bundle.yaml -n trident-protect
```

Crie um pacote de suporte usando a CLI

Passos

1. Crie o pacote de suporte, substituindo os valores entre colchetes pelas informações do seu ambiente.

O `trigger-type` determina se o pacote será criado imediatamente ou se o horário de criação será definido pelo agendamento, e pode ser `Manual` ou `Scheduled`. A configuração padrão é `Manual`.

Por exemplo:

```
tridentctl-protect create autosupportbundle <my-bundle-name>
--trigger-type <trigger-type> -n trident-protect
```

Monitore e recupere o pacote de suporte

Após criar um pacote de suporte usando qualquer um dos métodos, você pode monitorar o progresso da geração e recuperá-lo para seu sistema local.

Passos

1. Aguarde até que o `status.generationState` atinja o estado `Completed`. Você pode monitorar o progresso da geração com o seguinte comando:

   ```
   kubectl get autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

2. Recupere o pacote de suporte para o seu sistema local. Obtenha o comando de cópia do pacote AutoSupport concluído:

   ```
   kubectl describe autosupportbundle trident-protect-support-bundle -n trident-protect
   ```

   Localize o comando `kubectl cp` na saída e execute-o, substituindo o argumento de destino pelo diretório local de sua preferência.

Atualizar Trident Protect

Você pode atualizar Trident Protect para a versão mais recente para aproveitar novos recursos ou correções de bugs.

- Ao atualizar da versão 24.10, snapshots em execução durante a atualização podem falhar. Essa falha não impede que snapshots futuros, sejam manuais ou agendados, sejam criados. Se um snapshot falhar durante a atualização, você pode criar manualmente um novo snapshot para garantir que seu aplicativo esteja protegido.

  Para evitar possíveis falhas, você pode desativar todos os agendamentos de snapshots antes da atualização e reativá-los posteriormente. No entanto, isso resulta na perda de quaisquer snapshots agendados durante o período de atualização.

- Para instalações em registros privados, certifique-se de que o Helm chart e as imagens necessárias para a versão de destino estejam disponíveis em seu registro privado e verifique se seus valores Helm personalizados são compatíveis com a nova versão do chart. Para mais informações, consulte "Instale Trident Protect a partir de um registro privado".

Para atualizar Trident Protect, execute as seguintes etapas.

Passos

1. Atualize o repositório do Trident:

```
helm repo update
```

2. Atualize os CRDs do Trident Protect:

   Esta etapa é necessária se você estiver atualizando de uma versão anterior à 25.06, pois os CRDs agora estão incluídos no Helm chart do Trident Protect.

   a. Execute este comando para transferir o gerenciamento de CRDs de `trident-protect-crds` para `trident-protect`:

```
kubectl get crd | grep protect.trident.netapp.io | awk '{print $1}' |
xargs -I {} kubectl patch crd {} --type merge -p '{"metadata":
{"annotations":{"meta.helm.sh/release-name": "trident-protect"}}}'
```

   b. Execute este comando para excluir o segredo do Helm para o `trident-protect-crds` chart:

   Não desinstale o `trident-protect-crds` chart usando o Helm, pois isso pode remover seus CRDs e quaisquer dados relacionados.

```
kubectl delete secret -n trident-protect -l name=trident-protect-
crds,owner=helm
```

3. Atualizar Trident Protect:

```
helm upgrade trident-protect netapp-trident-protect/trident-protect
--version 100.2510.0 --namespace trident-protect
```

Você pode configurar o nível de registro durante a atualização adicionando `--set logLevel=debug` ao comando de atualização. O nível de registro padrão é `warn`. O registro de depuração é recomendado para solução de problemas, pois ajuda a equipe de suporte da NetApp a diagnosticar problemas sem exigir alterações no nível de registro ou reprodução do problema.

Gerenciar e proteger aplicativos

Use objetos do Trident Protect AppVault para gerenciar buckets

O recurso personalizado (CR) do bucket para Trident Protect é conhecido como um AppVault. Objetos AppVault são a representação declarativa do fluxo de trabalho do Kubernetes de um bucket de armazenamento. Um CR AppVault contém as configurações necessárias para que um bucket seja usado em operações de proteção, como backup, snapshots, operações de restauração e replicação SnapMirror. Somente administradores podem criar AppVaults.

Você precisa criar um AppVault CR manualmente ou pela linha de comando ao executar operações de proteção de dados em um aplicativo. O AppVault CR é específico para o seu ambiente, e você pode usar os exemplos desta página como guia ao criar AppVault CRs.

Certifique-se de que o AppVault CR esteja no cluster onde Trident Protect está instalado. Se o AppVault CR não existir ou você não conseguir acessá-lo, a linha de comando exibirá um erro.

Configurar autenticação e senhas do AppVault

Antes de criar um AppVault CR, certifique-se de que o AppVault e o data mover escolhido possam se autenticar com o provedor e quaisquer recursos relacionados.

Senhas do repositório do Data Mover

Ao criar objetos AppVault usando CRs ou o plugin CLI do Trident Protect, você pode especificar um segredo do Kubernetes com senhas personalizadas para criptografia Restic e Kopia. Se você não especificar um segredo, Trident Protect usará uma senha padrão.

- Ao criar manualmente CRs do AppVault, use o campo **spec.dataMoverPasswordSecretRef** para especificar o segredo.
- Ao criar objetos AppVault usando a Trident Protect CLI, utilize o argumento `--data-mover-password-secret-ref` para especificar o segredo.

Crie um segredo de senha do repositório de Data Mover

Use os exemplos a seguir para criar a senha secreta. Ao criar AppVault objetos, você pode instruir Trident Protect a usar essa senha secreta para autenticar com o repositório do data mover.

- Dependendo do gerenciador de dados que você estiver usando, você só precisa incluir a senha correspondente para esse gerenciador de dados. Por exemplo, se você estiver usando Restic e não planeja usar Kopia no futuro, você pode incluir apenas a senha do Restic ao criar o segredo.
- Guarde a senha em um local seguro. Você precisará dela para restaurar dados no mesmo cluster ou em um diferente. Se o cluster ou o `trident-protect` namespace for excluído, você não poderá restaurar seus backups ou snapshots sem a senha.

Use um CR

```
---
apiVersion: v1
data:
  KOPIA_PASSWORD: <base64-encoded-password>
  RESTIC_PASSWORD: <base64-encoded-password>
kind: Secret
metadata:
  name: my-optional-data-mover-secret
  namespace: trident-protect
type: Opaque
```

Use o CLI

```
kubectl create secret generic my-optional-data-mover-secret \
--from-literal=KOPIA_PASSWORD=<plain-text-password> \
--from-literal=RESTIC_PASSWORD=<plain-text-password> \
-n trident-protect
```

Permissões IAM de armazenamento compatível com S3

Ao acessar armazenamento compatível com S3, como Amazon S3, S3 genérico, "StorageGrid S3", ou "ONTAP S3" usando Trident Protect, você precisa garantir que as credenciais do usuário fornecidas tenham as permissões necessárias para acessar o bucket. O exemplo a seguir mostra uma política que concede as permissões mínimas necessárias para acesso com Trident Protect. Você pode aplicar essa política ao usuário que gerencia as políticas de buckets compatíveis com S3.

```
{
  "Version": "2012-10-17",
  "Statement": [
    {
      "Effect": "Allow",
      "Action": [
        "s3:PutObject",
        "s3:GetObject",
        "s3:ListBucket",
        "s3:DeleteObject"
      ],
      "Resource": "*"
    }
  ]
}
```

Para obter mais informações sobre as políticas do Amazon S3, consulte os exemplos em "Documentação do Amazon S3".

Identidade do Pod EKS para autenticação Amazon S3

Trident Protect oferece suporte à EKS Pod Identity para operações de movimentação de dados do Kopia. Esse recurso permite acesso seguro a buckets do S3 sem armazenar credenciais da AWS em segredos do Kubernetes.

Requisitos para EKS Pod Identity com Trident Protect

Antes de usar o EKS Pod Identity com Trident Protect, certifique-se do seguinte:

- Seu cluster EKS tem Pod Identity habilitado.
- Você criou uma função do IAM com as permissões necessárias para o bucket do S3. Para saber mais, consulte "Permissões IAM de armazenamento compatível com S3".
- A função IAM está associada às seguintes contas de serviço do Trident Protect:
  - `<trident-protect>-controller-manager`
  - `<trident-protect>-resource-backup`
  - `<trident-protect>-resource-restore`
  - `<trident-protect>-resource-delete`

Para obter instruções detalhadas sobre como habilitar a identidade do Pod e associar funções do IAM a contas de serviço, consulte o "Documentação do AWS EKS Pod Identity".

AppVault Configuração Ao usar o EKS Pod Identity, configure seu AppVault CR com a `useIAM: true` flag em vez de credenciais explícitas:

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: eks-protect-vault
  namespace: trident-protect
spec:
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-aws
      endpoint: s3.example.com
      useIAM: true
```

AppVault exemplos de geração de chaves para provedores de nuvem

Ao definir um AppVault CR, você precisa incluir credenciais para acessar os recursos hospedados pelo provedor, a menos que esteja usando autenticação IAM. A forma como você gera as chaves para as credenciais varia de acordo com o provedor. A seguir estão exemplos de geração de chaves pela linha de comando para vários provedores. Você pode usar os seguintes exemplos para criar chaves para as credenciais de cada provedor de nuvem.

Google Cloud

```
kubectl create secret generic <secret-name> \
--from-file=credentials=<mycreds-file.json> \
-n trident-protect
```

Amazon S3 (AWS)

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<amazon-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

Microsoft Azure

```
kubectl create secret generic <secret-name> \
--from-literal=accountKey=<secret-name> \
-n trident-protect
```

S3 genérico

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<generic-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

ONTAP S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<ontap-s3-trident-protect-src-bucket
-secret> \
-n trident-protect
```

StorageGrid S3

```
kubectl create secret generic <secret-name> \
--from-literal=accessKeyID=<objectstorage-accesskey> \
--from-literal=secretAccessKey=<storagegrid-s3-trident-protect-src
-bucket-secret> \
-n trident-protect
```

AppVault creation exemplos

A seguir estão exemplos de definições do AppVault para cada provedor.

AppVault CR exemplos

Você pode usar os seguintes exemplos de CR para criar objetos AppVault para cada provedor de nuvem.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia do repositório Restic e Kopia. Consulte Senhas do repositório do Data Mover para obter mais informações.
- Para objetos AppVault do Amazon S3 (AWS), você pode opcionalmente especificar um sessionToken, o que é útil se você estiver usando logon único (SSO) para autenticação. Esse token é criado quando você gera chaves para o provedor em AppVault exemplos de geração de chaves para provedores de nuvem.
- Para objetos S3 AppVault, você pode opcionalmente especificar uma URL de proxy de saída para o tráfego S3 de saída usando a chave `spec.providerConfig.S3.proxyURL`.

Google Cloud

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: gcp-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GCP
  providerConfig:
    gcp:
      bucketName: trident-protect-src-bucket
      projectID: project-id
  providerCredentials:
    credentials:
      valueFromSecret:
        key: credentials
        name: gcp-trident-protect-src-bucket-secret
```

Amazon S3 (AWS)

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: amazon-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: AWS
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
    sessionToken:
      valueFromSecret:
        key: sessionToken
        name: s3-secret
```

Para ambientes EKS usando Pod Identity com Kopia data mover, você pode remover a `providerCredentials` seção e adicionar `useIAM: true` sob a configuração `s3` em vez disso.

Microsoft Azure

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: azure-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: Azure
  providerConfig:
    azure:
      accountName: account-name
      bucketName: trident-protect-src-bucket
  providerCredentials:
    accountKey:
      valueFromSecret:
        key: accountKey
        name: azure-trident-protect-src-bucket-secret
```

S3 genérico

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: generic-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: GenericS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

ONTAP S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: ontap-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: OntapS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

StorageGrid S3

```
apiVersion: protect.trident.netapp.io/v1
kind: AppVault
metadata:
  name: storagegrid-s3-trident-protect-src-bucket
  namespace: trident-protect
spec:
  dataMoverPasswordSecretRef: my-optional-data-mover-secret
  providerType: StorageGridS3
  providerConfig:
    s3:
      bucketName: trident-protect-src-bucket
      endpoint: s3.example.com
      proxyURL: http://10.1.1.1:3128
  providerCredentials:
    accessKeyID:
      valueFromSecret:
        key: accessKeyID
        name: s3-secret
    secretAccessKey:
      valueFromSecret:
        key: secretAccessKey
        name: s3-secret
```

AppVault exemplos de criação usando o Trident Protect CLI

Você pode usar os seguintes exemplos de comandos da CLI para criar AppVault CRs para cada provedor.

- Opcionalmente, você pode especificar um segredo do Kubernetes que contenha senhas personalizadas para a criptografia do repositório Restic e Kopia. Consulte Senhas do repositório do Data Mover para obter mais informações.
- Para objetos S3 AppVault, você pode opcionalmente especificar uma URL de proxy de saída para o tráfego S3 de saída usando o `--proxy-url <ip_address:port>` argumento.

Google Cloud

```
tridentctl-protect create vault GCP <vault-name> \
--bucket <mybucket> \
--project <my-gcp-project> \
--secret <secret-name>/credentials \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Amazon S3 (AWS)

```
tridentctl-protect create vault AWS <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Microsoft Azure

```
tridentctl-protect create vault Azure <vault-name> \
--account <account-name> \
--bucket <bucket-name> \
--secret <secret-name> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

S3 genérico

```
tridentctl-protect create vault GenericS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

ONTAP S3

```
tridentctl-protect create vault OntapS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

StorageGrid S3

```
tridentctl-protect create vault StorageGridS3 <vault-name> \
--bucket <bucket-name> \
--secret  <secret-name>  \
--endpoint <s3-endpoint> \
--data-mover-password-secret-ref <my-optional-data-mover-secret> \
-n trident-protect
```

Opções de `providerConfig.s3` configuração suportadas

Consulte a tabela a seguir para as opções de configuração do provedor S3:

Parâmetro	Descrição	Padrão	Exemplo
`providerConfig.s3.skipCertValidation`	Desative a verificação de certificado SSL/TLS.	falso	"true", "false"
`providerConfig.s3.secure`	Habilite a comunicação HTTPS segura com o endpoint S3.	verdadeiro	"true", "false"
`providerConfig.s3.proxyURL`	Especifique a URL do servidor proxy usada para conectar-se ao S3.	Nenhum	http://proxy.example.com:8080
`providerConfig.s3.rootCA`	Forneça um certificado CA raiz personalizado para verificação SSL/TLS.	Nenhum	"CN=MyCustomCA"
`providerConfig.s3.useIAM`	Habilite a autenticação IAM para acessar buckets do S3. Aplicável para EKS Pod Identity.	falso	true, false

Ver informações do AppVault

Você pode usar o plugin Trident Protect CLI para visualizar informações sobre objetos AppVault que você criou no cluster.

Passos

1. Visualize o conteúdo de um objeto AppVault:

```
tridentctl-protect get appvaultcontent gcp-vault \
--show-resources all \
-n trident-protect
```

Exemplo de saída:

```
+-------------+-------+----------+----------------------------
+---------------------------+
|   CLUSTER   |  APP  |   TYPE   |            NAME             |
TIMESTAMP         |
+-------------+-------+----------+----------------------------
+---------------------------+
|             | mysql | snapshot | mysnap                      | 2024-
08-09 21:02:11 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815180300 | 2024-
08-15 18:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:06 (UTC) |
| production1 | mysql | snapshot | hourly-e7db6-20240815200300 | 2024-
08-15 20:03:06 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815180300 | 2024-
08-15 18:04:25 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815190300 | 2024-
08-15 19:03:30 (UTC) |
| production1 | mysql | backup   | hourly-e7db6-20240815200300 | 2024-
08-15 20:04:21 (UTC) |
| production1 | mysql | backup   | mybackup5                   | 2024-
08-09 22:25:13 (UTC) |
|             | mysql | backup   | mybackup                    | 2024-
08-09 21:02:52 (UTC) |
+-------------+-------+----------+----------------------------
+---------------------------+
```

2. Opcionalmente, para visualizar o AppVaultPath de cada recurso, use a flag `--show-paths`.

   O nome do cluster na primeira coluna da tabela só estará disponível se um nome de cluster tiver sido especificado na instalação do Trident Protect via helm. Por exemplo: `--set clusterName=production1`.

Remover um AppVault

Você pode remover um objeto AppVault a qualquer momento.

Não remova a chave `finalizers` no CR AppVault antes de excluir o objeto AppVault. Se você fizer isso, pode resultar em dados residuais no bucket AppVault e recursos órfãos no cluster.

Antes de começar

Certifique-se de ter excluído todos os CRs de snapshot e backup que estão sendo usados pelo AppVault que você deseja excluir.

Remova um AppVault usando a CLI do Kubernetes

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault a ser removido:

```
kubectl delete appvault <appvault-name> \
-n trident-protect
```

Remova um AppVault usando o Trident Protect CLI

1. Remova o objeto AppVault, substituindo `appvault-name` pelo nome do objeto AppVault a ser removido:

```
tridentctl-protect delete appvault <appvault-name> \
-n trident-protect
```

Defina uma aplicação para gestão com Trident Protect

Você pode definir um aplicativo que deseja gerenciar com Trident Protect criando um CR de aplicativo e um CR de AppVault associado.

Criar um AppVault CR

Você precisa criar um AppVault CR que será usado ao executar operações de proteção de dados no aplicativo, e o AppVault CR precisa residir no cluster onde Trident Protect está instalado. O AppVault CR é específico para o seu ambiente; para exemplos de AppVault CRs, consulte "Recursos personalizados do AppVault."

Definir uma aplicação

Você precisa definir cada aplicativo que deseja gerenciar com Trident Protect. Você pode definir um aplicativo para gerenciamento criando manualmente um CR do aplicativo ou usando a CLI do Trident Protect.

Adicionar um aplicativo usando um CR

Passos

1. Crie o arquivo CR do aplicativo de destino:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `maria-app.yaml`).
   b. Configurar os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para operações de proteção fazem referência a esse valor.
      - **spec.includedNamespaces**: (*Obrigatório*) Use o seletor de namespace e rótulo para especificar os namespaces e recursos que o aplicativo utiliza. O namespace do aplicativo deve fazer parte desta lista. O seletor de rótulo é opcional e pode ser usado para filtrar recursos dentro de cada namespace especificado.
      - **spec.includedClusterScopedResources**: (*Opcional*) Use este atributo para especificar recursos com escopo de cluster a serem incluídos na definição do aplicativo. Este atributo permite selecionar esses recursos com base em seu grupo, versão, tipo e rótulos.
        - **groupVersionKind**: (*Obrigatório*) Especifica o grupo de API, a versão e o tipo do recurso com escopo de cluster.
        - **labelSelector**: (Opcional) Filtra os recursos com escopo de cluster com base em seus rótulos.
      - **metadata.annotations.protect.trident.netapp.io/skip-vm-freeze**: (*Opcional*) Esta anotação só se aplica a aplicações definidas a partir de máquinas virtuais, como em KubeVirt ambientes, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Especifique se esta aplicação pode gravar no sistema de arquivos durante um snapshot. Se definida como true, a aplicação ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se definida como false, a aplicação ignora a configuração global e o sistema de arquivos é congelado durante um snapshot. Se especificada, mas a aplicação não tiver máquinas virtuais na definição da aplicação, a anotação será ignorada. Se não especificada, a aplicação segue a "configuração global de congelamento do Trident Protect".

Se você precisar aplicar essa anotação depois que um aplicativo já tiver sido criado, pode usar o seguinte comando:

```
kubectl annotate application -n <application CR namespace> <application CR name> protect.trident.netapp.io/skip-vm-freeze="true"
```

+
Exemplo YAML:

+

```
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  annotations:
    protect.trident.netapp.io/skip-vm-freeze: "false"
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: namespace-1
      labelSelector:
        matchLabels:
          app: example-app
    - namespace: namespace-2
      labelSelector:
        matchLabels:
          app: another-example-app
  includedClusterScopedResources:
    - groupVersionKind:
        group: rbac.authorization.k8s.io
        kind: ClusterRole
        version: v1
      labelSelector:
        matchLabels:
          mylabel: test
```

1. (Opcional) Adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:

    - **resourceFilter.resourceSelectionCriteria**: (obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:

        - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos dentro de cada elemento (group, kind, version) correspondem como uma operação AND.
            - **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
            - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
            - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
            - **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.

- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

> Quando ambos `resourceFilter` e `labelSelector` são usados, `resourceFilter` é executado primeiro e, em seguida, `labelSelector` é aplicado aos recursos resultantes.

Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

2. Após criar a CR do aplicativo para corresponder ao seu ambiente, aplique a CR. Por exemplo:

```
kubectl apply -f maria-app.yaml
```

Passos

1. Crie e aplique a definição do aplicativo usando um dos exemplos a seguir, substituindo os valores entre colchetes pelas informações do seu ambiente. Você pode incluir namespaces e recursos na definição do aplicativo usando listas separadas por vírgulas com os argumentos mostrados nos exemplos.

   Você pode, opcionalmente, usar uma anotação ao criar um aplicativo para especificar se o aplicativo pode gravar no sistema de arquivos durante um snapshot. Isso se aplica somente a aplicativos definidos a partir de máquinas virtuais, como em KubeVirt ambientes, onde o congelamento do sistema de arquivos ocorre antes dos snapshots. Se você definir a anotação como `true`, o aplicativo ignora a configuração global e pode gravar no sistema de arquivos durante um snapshot. Se você definir como `false`, o aplicativo ignora a configuração global e o sistema de arquivos é congelado durante um snapshot. Se você usar a anotação, mas o aplicativo não tiver máquinas virtuais na

definição do aplicativo, a anotação será ignorada. Se você não usar a anotação, o aplicativo seguirá a "configuração global de congelamento do Trident Protect".

Para especificar a anotação ao usar a CLI para criar um aplicativo, você pode usar a `--annotation` flag.

- Crie o aplicativo e utilize a configuração global para o comportamento de congelamento do sistema de arquivos:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace>
```

- Crie o aplicativo e configure a definição local do aplicativo para o comportamento de congelamento do sistema de arquivos:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-
namespace> --annotation protect.trident.netapp.io/skip-vm-freeze
=<"true"|"false">
```

Você pode usar `--resource-filter-include` e `--resource-filter-exclude` sinalizadores para incluir ou excluir recursos com base em `resourceSelectionCriteria` como grupo, tipo, versão, rótulos, nomes e namespaces, conforme mostrado no exemplo a seguir:

```
tridentctl-protect create application <my_new_app_cr_name>
--namespaces <namespaces_to_include> --csr
<cluster_scoped_resources_to_include> --namespace <my-app-namespace>
--resource-filter-include
'[{"Group":"apps","Kind":"Deployment","Version":"v1","Names":["my-
deployment"],"Namespaces":["my-
namespace"],"LabelSelectors":["app=my-app"]}]'
```

Proteja aplicativos usando Trident Protect

Você pode proteger todos os aplicativos gerenciados pelo Trident Protect tirando snapshots e backups usando uma política de proteção automatizada ou sob demanda.

Você pode configurar Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com Trident Protect".

Criar um snapshot sob demanda

Você pode criar um snapshot sob demanda a qualquer momento.

> Os recursos com escopo de cluster são incluídos em um backup, Snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie um instantâneo usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: o nome do aplicativo no Kubernetes para o qual será criado o Snapshot.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do Snapshot (metadados) deve ser armazenado.
   - **spec.reclaimPolicy**: (*Opcional*) Define o que acontece com o AppArchive de um snapshot quando o CR do snapshot é excluído. Isso significa que mesmo quando definido como `Retain`, o snapshot será excluído. Opções válidas:
     - `Retain` (padrão)
     - `Delete`

     ```
     apiVersion: protect.trident.netapp.io/v1
     kind: Snapshot
     metadata:
       namespace: my-app-namespace
       name: my-cr-name
     spec:
       applicationRef: my-application
       appVaultRef: appvault-name
       reclaimPolicy: Delete
     ```
3. Após preencher o arquivo `trident-protect-snapshot-cr.yaml` com os valores corretos, aplique a CR:

   ```
   kubectl apply -f trident-protect-snapshot-cr.yaml
   ```

Criar um instantâneo usando a CLI

Passos

1. Crie o instantâneo, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

   ```
   tridentctl-protect create snapshot <my_snapshot_name> --appvault
   <my_appvault_name> --app <name_of_app_to_snapshot> -n
   <application_namespace>
   ```

Crie um backup sob demanda

Você pode fazer backup de um app a qualquer momento.

> Os recursos com escopo de cluster são incluídos em um backup, Snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de backup do s3 de longa duração. Se o token expirar durante a operação de backup, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Criar um backup usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes a ser feito backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.dataMover**: (*Opcional*) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia` (padrão)
   - **spec.reclaimPolicy**: (*Opcional*) Define o que acontece com um backup quando liberado de sua reivindicação. Valores possíveis:
     - `Delete`
     - `Retain` (padrão)
   - **spec.snapshotRef**: (*Opcional*): Nome do snapshot a ser usado como origem do backup. Se não for fornecido, um snapshot temporário será criado e feito backup.

     Exemplo YAML:

```yaml
---
apiVersion: protect.trident.netapp.io/v1
kind: Backup
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  dataMover: Kopia
```

3. Após preencher o arquivo `trident-protect-backup-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-cr.yaml
```

Criar um backup usando a CLI

Passos

1. Crie o backup, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

```
tridentctl-protect create backup <my_backup_name> --appvault <my-vault-name> --app <name_of_app_to_back_up> --data-mover <Kopia_or_Restic> -n <application_namespace>
```

Você pode opcionalmente usar a `--full-backup` flag para especificar se um backup deve ser não incremental. Por padrão, todos os backups são incrementais. Quando essa flag é usada, o backup se torna não incremental. A melhor prática é realizar um backup completo periodicamente e depois realizar backups incrementais entre os backups completos para minimizar o risco associado a restaurações.

Anotações de backup suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar um backup CR:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/backup-completo	string	Especifica se um backup deve ser não incremental. Defina como `true` para criar um backup não incremental. A melhor prática é realizar um backup completo periodicamente e, em seguida, realizar backups incrementais entre os backups completos para minimizar o risco associado às restaurações.	"false"
protect.trident.netapp.io/snapshot-completion-timeout	string	O tempo máximo permitido para a conclusão geral da operação de Snapshot.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	string	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	string	O tempo máximo permitido para que snapshots de volume sejam criados.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe.	"1200" (20 minutos)

Crie um cronograma de proteção de dados

Uma política de proteção protege um app criando snapshots, backups ou ambos em uma programação definida. Você pode optar por criar snapshots e backups a cada hora, dia, semana e mês, e pode especificar o número de cópias a reter. Você pode agendar um backup completo não incremental usando a anotação full-backup-rule. Por padrão, todos os backups são incrementais. Realizar um backup completo periodicamente, juntamente com backups incrementais entre eles, ajuda a reduzir o risco associado a restaurações.

- Você pode criar agendamentos para snapshots apenas definindo `backupRetention` como zero e `snapshotRetention` como um valor maior que zero. Definir `snapshotRetention` como zero significa que qualquer backup agendado ainda criará snapshots, mas estes são temporários e são excluídos imediatamente após a conclusão do backup.
- Os recursos com escopo de cluster são incluídos em um backup, Snapshot ou clone se forem explicitamente referenciados na definição do aplicativo ou se tiverem referências a qualquer um dos namespaces do aplicativo.

Crie um cronograma usando uma CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-schedule-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.dataMover**: (*Opcional*) Uma string indicando qual ferramenta de backup usar para a operação de backup. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Restic`
     - `Kopia` (padrão)
   - **spec.applicationRef**: o nome do aplicativo no Kubernetes para o qual será feito o backup.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup deve ser armazenado.
   - **spec.backupRetention**: (*Obrigatório*) O número de backups a serem mantidos. Zero indica que nenhum backup deve ser criado (apenas snapshots).
   - **spec.backupReclaimPolicy**: (*Opcional*) Determina o que acontece com um backup se o backup CR for excluído durante seu período de retenção. Após o período de retenção, os backups são sempre excluídos. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain` (padrão)
     - `Delete`
   - **spec.snapshotRetention**: (*Obrigatório*) O número de snapshots a serem mantidos. Zero indica que nenhum snapshot deve ser criado.
   - **spec.snapshotReclaimPolicy**: (*Opcional*) Determina o que acontece com um snapshot se o CR do snapshot for excluído durante seu período de retenção. Após o período de retenção, os snapshots são sempre excluídos. Valores possíveis (diferencia maiúsculas de minúsculas):
     - `Retain`
     - `Delete` (padrão)
   - **spec.granularity**: A frequência com que o agendamento deve ser executado. Valores possíveis, juntamente com os campos associados obrigatórios:
     - `Hourly` (requer que você especifique `spec.minute`)
     - `Daily` (requer que você especifique `spec.minute` e `spec.hour`)
     - `Weekly` (requer que você especifique `spec.minute, spec.hour`, e `spec.dayOfWeek`)
     - `Monthly` (requer que você especifique `spec.minute, spec.hour`, e `spec.dayOfMonth`)
     - `Custom`
   - **spec.dayOfMonth**: (opcional) O dia do mês (1 - 31) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Monthly`. O valor deve ser fornecido como uma string.
   - **spec.dayOfWeek**: (opcional) O dia da semana (0 - 7) em que a programação deve ser executada. Valores de 0 ou 7 indicam domingo. Este campo é obrigatório se a granularidade

estiver definida como `Weekly`. O valor deve ser fornecido como uma string.

- **spec.hour**: (Opcional) A hora do dia (0 - 23) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Daily`, `Weekly`, ou `Monthly`. O valor deve ser fornecido como uma string.
- **spec.minute**: (Opcional) O minuto da hora (0 - 59) em que a programação deve ser executada. Este campo é obrigatório se a granularidade estiver definida como `Hourly`, `Daily`, `Weekly` ou `Monthly`. O valor deve ser fornecido como uma string.

Exemplo de YAML para backup e agendamento de snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-cr-name
spec:
  dataMover: Kopia
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "15"
  snapshotRetention: "15"
  granularity: Daily
  hour: "0"
  minute: "0"
```

Exemplo de YAML para agendamento somente com snapshot:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  namespace: my-app-namespace
  name: my-snapshot-schedule
spec:
  applicationRef: my-application
  appVaultRef: appvault-name
  backupRetention: "0"
  snapshotRetention: "15"
  granularity: Daily
  hour: "2"
  minute: "0"
```

3. Após preencher o arquivo `trident-protect-schedule-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-schedule-cr.yaml
```

Crie um agendamento usando a linha de comando (CLI)

Passos

1. Crie o cronograma de proteção, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

> Você pode usar `tridentctl-protect create schedule --help` para visualizar informações de ajuda detalhadas para este comando.

```
tridentctl-protect create schedule <my_schedule_name> \
  --appvault <my_appvault_name> \
  --app <name_of_app_to_snapshot> \
  --backup-retention <how_many_backups_to_retain> \
  --backup-reclaim-policy <Retain|Delete (default Retain)> \
  --data-mover <Kopia_or_Restic> \
  --day-of-month <day_of_month_to_run_schedule> \
  --day-of-week <day_of_week_to_run_schedule> \
  --granularity <frequency_to_run> \
  --hour <hour_of_day_to_run> \
  --minute <minute_of_hour_to_run> \
  --recurrence-rule <recurrence> \
  --snapshot-retention <how_many_snapshots_to_retain> \
  --snapshot-reclaim-policy <Retain|Delete (default Delete)> \
  --full-backup-rule <string> \
  --run-immediately <true|false> \
  -n <application_namespace>
```

As seguintes opções oferecem controle adicional sobre sua programação:

- **Agendamento de backup completo**: Use a `--full-backup-rule` flag para agendar backups completos não incrementais. Esta flag funciona apenas com `--granularity Daily`. Valores possíveis:
  - `Always`: Faça um backup completo todos os dias.
  - Dias da semana específicos: especifique um ou mais dias separados por vírgulas (por exemplo, `"Monday,Thursday"`). Valores válidos: segunda-feira, terça-feira, quarta-feira, quinta-feira, sexta-feira, sábado, domingo.

> A `--full-backup-rule`opção não funciona com granularidade horária, semanal ou mensal.

- **Agendamentos somente de instantâneo**: Defina `--backup-retention 0` e especifique um valor maior que zero para `--snapshot-retention`.

Anotações de cronograma suportadas

A tabela a seguir descreve as anotações que você pode usar ao criar um schedule CR:

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/full-backup-rule	string	Especifica a regra para agendamento de backups completos. Você pode configurá-la para `Always` backup completo ou personalizá-la de acordo com suas necessidades. Por exemplo, se você escolher granularidade diária, poderá especificar os dias da semana em que o backup completo deve ocorrer (por exemplo, `"Monday,Thursday"`). Os valores válidos para os dias da semana são: Monday, Tuesday, Wednesday, Thursday, Friday, Saturday, Sunday. Observe que esta anotação só pode ser usada com agendamentos que tenham `granularity` definido como `Daily`.	Não configurado (todos os backups são incrementais)
protect.trident.netapp.io/snapshot-completion-timeout	string	O tempo máximo permitido para a conclusão geral da operação de Snapshot.	"60m"
protect.trident.netapp.io/volume-snapshots-ready-to-use-timeout	string	Tempo máximo permitido para que os snapshots de volume atinjam o estado pronto para uso.	"30m"
protect.trident.netapp.io/volume-snapshots-created-timeout	string	O tempo máximo permitido para que snapshots de volume sejam criados.	"5m"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe.	"1200" (20 minutos)

Excluir um instantâneo

Exclua os snapshots agendados ou sob demanda que você não precisa mais.

Passos

1. Remova o snapshot CR associado ao snapshot:

```
kubectl delete snapshot <snapshot_name> -n my-app-namespace
```

Excluir um backup

Exclua os backups agendados ou sob demanda que você não precisa mais.

Certifique-se de que a política de recuperação esteja configurada para `Delete` remover todos os dados de backup do storage de objetos. A configuração padrão da política é `Retain` para evitar perda de dados. Se a política não for alterada para `Delete`, os dados de backup permanecerão no storage de objetos e precisarão ser excluídos manualmente.

Passos

1. Remova o backup CR associado ao backup:

```
kubectl delete backup <backup_name> -n my-app-namespace
```

Verifique o status de uma operação de backup

Você pode usar a linha de comando para verificar o status de uma operação de backup que está em andamento, foi concluída ou falhou.

Passos

1. Utilize o seguinte comando para recuperar o status da operação de backup, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
kubectl get backup -n <namespace_name> <my_backup_cr_name> -o jsonpath
='{.status}'
```

Habilite backup e restauração para operações do azure-netapp-files (ANF)

Se você instalou Trident Protect, pode habilitar a funcionalidade de backup e restauração com uso eficiente de espaço para back-ends de armazenamento que utilizam a classe de armazenamento azure-netapp-files e foram criados antes do Trident 24.06. Essa funcionalidade funciona com volumes NFSv4 e não consome espaço adicional do pool de capacidade.

Antes de começar

Certifique-se do seguinte:

- Você instalou Trident Protect.
- Você definiu um aplicativo no Trident Protect. Este aplicativo terá funcionalidade de proteção limitada até que você conclua este procedimento.
- Você selecionou `azure-netapp-files` como classe de armazenamento padrão para seu backend de armazenamento.

Expandir para ver as etapas de configuração

1. Execute o seguinte no Trident se o volume ANF foi criado antes da atualização para Trident 24.10:

   a. Habilite o diretório de instantâneos para cada PV baseado em azure-netapp-files e associado à aplicação:

   ```
   tridentctl update volume <pv name> --snapshot-dir=true -n trident
   ```

   b. Confirme se o diretório de snapshot foi habilitado para cada PV associado:

   ```
   tridentctl get volume <pv name> -n trident -o yaml | grep
   snapshotDir
   ```

   Resposta:

   ```
   snapshotDirectory: "true"
   ```

   +
   Quando o diretório de snapshots não está habilitado, Trident Protect escolhe a funcionalidade de backup regular, que consome temporariamente espaço no pool de capacidade durante o processo de backup. Nesse caso, certifique-se de que haja espaço suficiente disponível no pool de capacidade para criar um volume temporário do tamanho do volume que está sendo feito backup.

Resultado

O aplicativo está pronto para backup e restauração usando Trident Protect. Cada PVC também está disponível para ser usado por outros aplicativos para backup e restauração.

Restaurar aplicativos

Restaure aplicativos usando Trident Protect

Você pode usar Trident Protect para restaurar seu aplicativo a partir de um snapshot ou backup. Restaurar a partir de um snapshot existente será mais rápido ao restaurar o aplicativo para o mesmo cluster.

- Ao restaurar um aplicativo, todos os ganchos de execução configurados para o aplicativo são restaurados juntamente com o aplicativo. Se houver um gancho de execução pós-restauração, ele é executado automaticamente como parte da operação de restauração.
- A restauração a partir de um backup para um namespace diferente ou para o namespace original é suportada para volumes qtree. No entanto, a restauração a partir de um snapshot para um namespace diferente ou para o namespace original não é suportada para volumes qtree.
- Você pode usar configurações avançadas para personalizar as operações de restauração. Para saber mais, consulte "Use as configurações avançadas de restauração do Trident Protect".

Restaurar a partir de um backup para um namespace diferente

Ao restaurar um backup para um namespace diferente usando um BackupRestore CR, Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger o aplicativo restaurado, crie backups ou snapshots sob demanda, ou estabeleça um cronograma de proteção.

- Restaurar um backup para um namespace diferente com recursos existentes não alterará nenhum recurso que compartilhe nomes com aqueles no backup. Para restaurar todos os recursos do backup, exclua e recrie o namespace de destino ou restaure o backup para um novo namespace.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente somente quando se utiliza a CLI.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` pelas informações do seu ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: BackupRestore
     metadata:
       name: my-cr-name
       namespace: my-destination-namespace
     spec:
       appArchivePath: my-backup-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > Trident Protect seleciona alguns recursos automaticamente devido à sua relação com os recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, Trident Protect também restaurará o pod associado.

   - **resourceFilter.resourceSelectionCriteria**: (obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos

dentro de cada elemento (group, kind, version) correspondem como uma operação AND.

- **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
- **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Após preencher o arquivo `trident-protect-backup-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use o CLI

Passos

1. Restaure o backup para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente. O `namespace-mapping` argumento usa namespaces separados por dois pontos para mapear os namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`. Por exemplo:

```
tridentctl-protect create backuprestore <my_restore_name> \
--backup <backup_namespace>/<backup_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restaurar de um backup para o namespace original

Você pode restaurar um backup para o namespace original a qualquer momento.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

> ⓘ Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-ipr-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.

     Por exemplo:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: BackupInplaceRestore
metadata:
  name: my-cr-name
  namespace: my-app-namespace
spec:
  appArchivePath: my-backup-path
  appVaultRef: appvault-name
```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > Trident Protect seleciona alguns recursos automaticamente devido à sua relação com os recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, Trident Protect também restaurará o pod associado.

   - **resourceFilter.resourceSelectionCriteria**: (obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos dentro de cada elemento (group, kind, version) correspondem como uma operação AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.

- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Após preencher o arquivo `trident-protect-backup-ipr-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-ipr-cr.yaml
```

Use o CLI

Passos

1. Restaure o backup para o namespace original, substituindo os valores entre colchetes pelas informações do seu ambiente. O `backup` argumento usa um namespace e um nome de backup no formato `<namespace>/<name>`. Por exemplo:

```
tridentctl-protect create backupinplacerestore <my_restore_name> \
--backup <namespace/backup_to_restore> \
-n <application_namespace>
```

Restaurar de um backup para um cluster diferente

Você pode restaurar um backup em um cluster diferente se houver um problema com o cluster original.

- Ao restaurar backups usando Kopia como o data mover, você pode opcionalmente especificar anotações no CR ou usando a CLI para controlar o comportamento do armazenamento temporário usado pelo Kopia. Consulte o "Documentação Kopia" para mais informações sobre as opções que você pode configurar. Use o comando `tridentctl-protect create --help` para mais informações sobre como especificar anotações com a Trident Protect CLI.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente somente quando se utiliza a CLI.

Antes de começar

Certifique-se de que os seguintes pré-requisitos sejam atendidos:

- O cluster de destino tem Trident Protect instalado.
- O cluster de destino tem acesso ao caminho do bucket do mesmo AppVault que o cluster de origem, onde o backup está armazenado.
- Certifique-se de que seu ambiente local possa se conectar ao bucket de storage de objetos definido no AppVault CR ao executar o comando `tridentctl-protect get appvaultcontent`. Se as restrições de rede impedirem o acesso, execute o Trident Protect CLI a partir de um pod no cluster de destino.
- Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.
  - Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
  - Consulte "Documentação da AWS" para obter mais informações sobre credenciais com recursos da AWS.

Passos

1. Verifique a disponibilidade do AppVault CR no cluster de destino usando o plugin Trident Protect CLI:

```
tridentctl-protect get appvault --context <destination_cluster_name>
```

   Certifique-se de que o namespace destinado à restauração do aplicativo exista no cluster de destino.

2. Visualize o conteúdo do backup disponível AppVault do cluster de destino:

```
tridentctl-protect get appvaultcontent <appvault_name> \
--show-resources backup \
--show-paths \
--context <destination_cluster_name>
```

Executar este comando exibe os backups disponíveis no AppVault, incluindo seus clusters de origem, nomes de aplicativos correspondentes, carimbos de data/hora e caminhos de arquivamento.

Exemplo de saída:

```
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
|   CLUSTER   |    APP    |  TYPE  |      NAME       |        TIMESTAMP
|    PATH     |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
| production1 | wordpress | backup | wordpress-bkup-1| 2024-10-30
08:37:40 (UTC)| backuppath1 |
| production1 | wordpress | backup | wordpress-bkup-2| 2024-10-30
08:37:40 (UTC)| backuppath2 |
+-------------+-----------+--------+-----------------
+--------------------------+-------------+
```

3. Restaure o aplicativo no cluster de destino usando o nome AppVault e o caminho do arquivo:

Use um CR

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-backup-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o
     jsonpath='{.status.appArchivePath}'
     ```

     Se o BackupRestore CR não estiver disponível, você pode usar o comando mencionado na etapa 2 para visualizar o conteúdo do backup.

   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` pelas informações do seu ambiente.

     Por exemplo:

```
apiVersion: protect.trident.netapp.io/v1
kind: BackupRestore
metadata:
  name: my-cr-name
  namespace: my-destination-namespace
spec:
  appVaultRef: appvault-name
  appArchivePath: my-backup-path
  namespaceMapping: [{"source": "my-source-namespace", "
destination": "my-destination-namespace"}]
```

3. Após preencher o arquivo `trident-protect-backup-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-backup-restore-cr.yaml
```

Use o CLI

1. Use o seguinte comando para restaurar o aplicativo, substituindo os valores entre colchetes pelas informações do seu ambiente. O argumento namespace-mapping usa namespaces separados por

dois pontos para mapear os namespaces de origem para os namespaces de destino corretos no formato source1:dest1,source2:dest2. Por exemplo:

```
tridentctl-protect create backuprestore <restore_name> \
--namespace-mapping <source_to_destination_namespace_mapping> \
--appvault <appvault_name> \
--path <backup_path> \
--context <destination_cluster_name> \
-n <application_namespace>
```

Restaurar a partir de um Snapshot para um namespace diferente

Você pode restaurar dados de um snapshot usando um arquivo de recurso personalizado (CR) para um namespace diferente ou para o namespace de origem original. Ao restaurar um snapshot para um namespace diferente usando um SnapshotRestore CR, Trident Protect restaura o aplicativo em um novo namespace e cria um CR de aplicativo para o aplicativo restaurado. Para proteger o aplicativo restaurado, crie backups ou snapshots sob demanda, ou estabeleça um agendamento de proteção.

- SnapshotRestore é compatível com o atributo `spec.storageClassMapping`, mas somente quando as classes de armazenamento de origem e destino usam o mesmo backend de armazenamento. Se você tentar restaurar para uma `StorageClass` que usa um backend de armazenamento diferente, a operação de restauração falhará.
- Ao usar uma CR para restaurar para um novo namespace, você deve criar manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente somente quando se utiliza a CLI.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get snapshots <SNAPHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` pelas informações do seu ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
       namespaceMapping: [{"source": "my-source-namespace", "destination": "my-destination-namespace"}]
     ```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > Trident Protect seleciona alguns recursos automaticamente devido à sua relação com os recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, Trident Protect também restaurará o pod associado.

   - **resourceFilter.resourceSelectionCriteria**: (obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos

dentro de cada elemento (group, kind, version) correspondem como uma operação AND.

- **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
- **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
- **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.
- **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Após preencher o arquivo `trident-protect-snapshot-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Use o CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>`.
   - O `namespace-mapping`argumento usa namespaces separados por dois pontos

para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2.

Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
--namespace-mapping <source_to_destination_namespace_mapping> \
-n <application_namespace>
```

Restaurar de um Snapshot para o namespace original

Você pode restaurar um snapshot para o namespace original a qualquer momento.

Antes de começar

Certifique-se de que o tempo de expiração do token de sessão da AWS seja suficiente para quaisquer operações de restauração do s3 de longa duração. Se o token expirar durante a operação de restauração, a operação pode falhar.

- Consulte o "Documentação do AWS API" para mais informações sobre como verificar a expiração do token de sessão atual.
- Consulte "Documentação do AWS IAM" para obter mais informações sobre credenciais com recursos da AWS.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-ipr-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get snapshots <SNAPSHOT_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotInplaceRestore
     metadata:
       name: my-cr-name
       namespace: my-app-namespace
     spec:
       appVaultRef: appvault-name
       appArchivePath: my-snapshot-path
     ```

3. (*Opcional*) Se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtros que incluam ou excluam recursos marcados com rótulos específicos:

   > Trident Protect seleciona alguns recursos automaticamente devido à sua relação com os recursos que você seleciona. Por exemplo, se você selecionar um recurso de reivindicação de volume persistente e ele tiver um pod associado, Trident Protect também restaurará o pod associado.

   - **resourceFilter.resourceSelectionCriteria**: (obrigatório para filtragem) Use `Include` ou `Exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos dentro de cada elemento (group, kind, version) correspondem como uma operação AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
       - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.

- **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "Include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Após preencher o arquivo `trident-protect-snapshot-ipr-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-ipr-cr.yaml
```

Use o CLI

Passos

1. Restaure o snapshot para o namespace original, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

```
tridentctl-protect create snapshotinplacerestore <my_restore_name> \
--snapshot <namespace/snapshot_to_restore> \
-n <application_namespace>
```

Verifique o status de uma operação de restauração

Você pode usar a linha de comando para verificar o status de uma operação de restauração que está em andamento, foi concluída ou falhou.

Passos

1. Use o seguinte comando para recuperar o status da operação de restauração, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
kubectl get backuprestore -n <namespace_name> <my_restore_cr_name> -o
jsonpath='{.status}'
```

Use as configurações avançadas de restauração do Trident Protect

Você pode personalizar as operações de restauração usando configurações avançadas, como anotações, configurações de namespace e opções de armazenamento para atender às suas necessidades específicas.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, os rótulos e anotações no namespace de destino são ajustados para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa Red Hat OpenShift, é importante observar o papel crucial das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados sigam as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o "OpenShift security context constraints documentação".

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configurar configurações adicionais do helm chart do Trident Protect".

Se você instalou o aplicativo de origem usando Helm com a `--create-namespace` flag, um tratamento especial é dado à chave de rótulo `name`. Durante o processo de restauração ou failover, Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele é copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e um de destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação, e como as anotações e os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino do exemplo antes da operação de restauração ou failover:

Espaço de nomes	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true"	• ambiente=produção • compliance=hipaa • name=ns-1
Espaço de nomes ns-2 (destino)	• annotation.one/key: "true" • anotação.three/chave: "falso"	• role=database

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, outras foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true" • anotação.three/chave: "falso"	• name=ns-2 • compliance=hipaa • ambiente=produção • role=database

Campos suportados

Esta seção descreve os campos adicionais disponíveis para operações de restauração.

Mapeamento de classe de armazenamento

O `spec.storageClassMapping` atributo define um mapeamento de uma classe de armazenamento presente na aplicação de origem para uma nova classe de armazenamento no cluster de destino. Você pode usar isso ao migrar aplicações entre clusters com classes de armazenamento diferentes ou ao alterar o backend de armazenamento para operações de BackupRestore.

Exemplo:

```
storageClassMapping:
  - destination: "destinationStorageClass1"
    source: "sourceStorageClass1"
  - destination: "destinationStorageClass2"
    source: "sourceStorageClass2"
```

Anotações suportadas

Esta seção lista as anotações suportadas para configurar diversos comportamentos no sistema. Se uma anotação não for definida explicitamente pelo usuário, o sistema usará o valor padrão.

Anotação	Tipo	Descrição	Valor padrão
protect.trident.netapp.io/data-mover-timeout-sec	string	O tempo máximo (em segundos) permitido para a operação de movimentação de dados ficar parada.	"300"
protect.trident.netapp.io/kopia-content-cache-size-limit-mb	string	O limite máximo de tamanho (em megabytes) para o cache de conteúdo do Kopia.	"1000"
protect.trident.netapp.io/pvc-bind-timeout-sec	string	Tempo máximo (em segundos) de espera para que qualquer PersistentVolumeClaims (PVC) recém-criado atinja a `Bound` fase antes que a operação falhe. Aplica-se a todos os tipos de CR de restauração (BackupRestore, BackupInplaceRestore, SnapshotRestore, SnapshotInplaceRestore). Use um valor maior se o seu backend de armazenamento ou cluster exigir mais tempo com frequência.	"1200" (20 minutos)

Replique aplicações usando NetApp SnapMirror e Trident Protect

Com Trident Protect, você pode usar os recursos de replicação assíncrona da tecnologia NetApp SnapMirror para replicar dados e alterações de aplicativos de um backend de armazenamento para outro, no mesmo cluster ou entre clusters diferentes.

Anotações e rótulos de namespace durante operações de restauração e failover

Durante as operações de restauração e failover, os rótulos e anotações no namespace de destino são ajustados para corresponder aos rótulos e anotações no namespace de origem. Rótulos ou anotações do namespace de origem que não existem no namespace de destino são adicionados, e quaisquer rótulos ou anotações já existentes são sobrescritos para corresponder ao valor do namespace de origem. Rótulos ou anotações que existem apenas no namespace de destino permanecem inalterados.

> Se você usa Red Hat OpenShift, é importante observar o papel crucial das anotações de namespace em ambientes OpenShift. As anotações de namespace garantem que os pods restaurados sigam as permissões e configurações de segurança apropriadas definidas pelas restrições de contexto de segurança (SCCs) do OpenShift e possam acessar volumes sem problemas de permissão. Para mais informações, consulte o "OpenShift security context constraints documentação".

Você pode impedir que anotações específicas no namespace de destino sejam sobrescritas definindo a variável de ambiente do Kubernetes `RESTORE_SKIP_NAMESPACE_ANNOTATIONS` antes de executar a operação de restauração ou failover. Por exemplo:

```
helm upgrade trident-protect -n trident-protect netapp-trident-
protect/trident-protect \
  --set-string
restoreSkipNamespaceAnnotations="{<annotation_key_to_skip_1>,<annotation_k
ey_to_skip_2>}" \
  --reuse-values
```

> Ao executar uma operação de restauração ou failover, quaisquer anotações e rótulos de namespace especificados em `restoreSkipNamespaceAnnotations` e `restoreSkipNamespaceLabels` são excluídos da operação de restauração ou failover. Certifique-se de que essas configurações sejam definidas durante a instalação inicial do Helm. Para saber mais, consulte "Configurar configurações adicionais do helm chart do Trident Protect".

Se você instalou o aplicativo de origem usando Helm com a `--create-namespace` flag, um tratamento especial é dado à chave de rótulo `name`. Durante o processo de restauração ou failover, Trident Protect copia esse rótulo para o namespace de destino, mas atualiza o valor para o valor do namespace de destino se o valor da origem corresponder ao namespace de origem. Se esse valor não corresponder ao namespace de origem, ele é copiado para o namespace de destino sem alterações.

Exemplo

O exemplo a seguir apresenta um namespace de origem e um de destino, cada um com anotações e rótulos diferentes. Você pode ver o estado do namespace de destino antes e depois da operação, e como as anotações e os rótulos são combinados ou sobrescritos no namespace de destino.

Antes da operação de restauração ou failover

A tabela a seguir ilustra o estado dos namespaces de origem e destino do exemplo antes da operação de restauração ou failover:

Espaço de nomes	Anotações	Etiquetas
Namespace ns-1 (fonte)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true"	• ambiente=produção • compliance=hipaa • name=ns-1
Espaço de nomes ns-2 (destino)	• annotation.one/key: "true" • anotação.three/chave: "falso"	• role=database

Após a operação de restauração

A tabela a seguir ilustra o estado do namespace de destino de exemplo após a operação de restauração ou failover. Algumas chaves foram adicionadas, outras foram sobrescritas e o `name` rótulo foi atualizado para corresponder ao namespace de destino:

Espaço de nomes	Anotações	Etiquetas
Espaço de nomes ns-2 (destino)	• annotation.one/key: "valoratualizado" • anotação.dois/chave: "true" • anotação.three/chave: "falso"	• name=ns-2 • compliance=hipaa • ambiente=produção • role=database

Você pode configurar Trident Protect para congelar e descongelar sistemas de arquivos durante operações de proteção de dados. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com Trident Protect".

Ganchos de execução durante failover e operações de reversão

Ao usar o relacionamento AppMirror para proteger sua aplicação, existem comportamentos específicos relacionados aos ganchos de execução que você deve conhecer durante operações de failover e reversão.

- Durante o failover, os hooks de execução são copiados automaticamente do cluster de origem para o cluster de destino. Você não precisa recriá-los manualmente. Após o failover, os hooks de execução estão presentes na aplicação e serão executados durante quaisquer ações relevantes.
- Durante a operação reversa ou ressincronização reversa, todos os ganchos de execução existentes no aplicativo são removidos. Quando o aplicativo de origem se torna o aplicativo de destino, esses ganchos de execução perdem a validade e são excluídos para impedir sua execução.

Para saber mais sobre ganchos de execução, consulte "Gerenciar os ganchos de execução do Trident Protect".

Estabeleça uma relação de replicação

Configurar uma relação de replicação envolve o seguinte:

- Escolher com que frequência você quer que o Trident Protect tire um snapshot do aplicativo (que inclui os recursos do Kubernetes do aplicativo, bem como os snapshots de volume para cada um dos volumes do aplicativo)

- Escolhendo o agendamento de replicação (inclui recursos do Kubernetes, bem como dados de volume persistente)
- Definindo o horário para a criação do snapshot

Passos

1. No cluster de origem, crie um AppVault para o aplicativo de origem. Dependendo do seu provedor de armazenamento, modifique um exemplo em "Recursos personalizados do AppVault" para adequá-lo ao seu ambiente:

Crie um AppVault usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-appvault-primary-source.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
- **spec.providerConfig**: (*Obrigatório*) Armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um bucketName e quaisquer outros detalhes necessários para o seu provedor. Anote os valores escolhidos, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esses valores. Consulte "Recursos personalizados do AppVault" para exemplos de CRs do AppVault com outros provedores.
- **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credencial necessárias para acessar o AppVault usando o provedor especificado.
  - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial deve vir de um segredo.
    - **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionada.
    - **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.
  - **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **name** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.
- **spec.providerType**: (*Obrigatório*) Determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Após preencher o arquivo `trident-protect-appvault-primary-source.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-appvault-primary-source.yaml -n
trident-protect
```

Crie um AppVault usando a CLI

a. Crie o AppVault, substituindo os valores entre colchetes por informações do seu ambiente:

```
tridentctl-protect create vault Azure <vault-name> --account
<account-name> --bucket <bucket-name> --secret <secret-name> -n
trident-protect
```

2. No cluster de origem, crie o CR do aplicativo de origem:

Crie o aplicativo de origem usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-app-source.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado do aplicativo. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
- **spec.includedNamespaces**: (*Obrigatório*) Uma matriz de namespaces e rótulos associados. Use nomes de namespace e, opcionalmente, restrinja o escopo dos namespaces com rótulos para especificar recursos que existem nos namespaces listados aqui. O namespace do aplicativo deve fazer parte desta matriz.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Application
metadata:
  name: my-app-name
  namespace: my-app-namespace
spec:
  includedNamespaces:
    - namespace: my-app-namespace
      labelSelector: {}
```

c. Após preencher o arquivo `trident-protect-app-source.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-app-source.yaml -n my-app-
namespace
```

Crie o aplicativo de origem usando a CLI

a. Crie a aplicação de origem. Por exemplo:

```
tridentctl-protect create app <my-app-name> --namespaces
<namespaces-to-be-included> -n <my-app-namespace>
```

3. Opcionalmente, no cluster de origem, tire um snapshot do aplicativo de origem. Esse snapshot é usado como base para o aplicativo no cluster de destino. Se você ignorar esta etapa, precisará aguardar a próxima execução de snapshot agendado para ter um snapshot recente. Para criar um snapshot sob demanda, consulte "Criar um snapshot sob demanda".

4. No cluster de origem, crie o CR de agendamento de replicação:

Além da programação fornecida abaixo, recomenda-se criar uma programação diária de snapshots separada, com um período de retenção de 7 dias, para manter um snapshot comum entre clusters ONTAP interligados. Isso garante que os snapshots estejam disponíveis por até 7 dias, mas o período de retenção pode ser personalizado de acordo com as necessidades do usuário.

Em caso de failover, o sistema pode usar esses snapshots por até 7 dias para operações de reversão. Essa abordagem torna o processo de reversão mais rápido e eficiente porque apenas as alterações feitas desde o último snapshot serão transferidas, não todos os dados.

Se um cronograma existente para o aplicativo já atender aos requisitos de retenção desejados, não serão necessários cronogramas adicionais.

Crie a programação de replicação usando uma CR

a. Crie um agendamento de replicação para o aplicativo de origem:

i. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-schedule.yaml`).

ii. Configurar os seguintes atributos:

- **metadata.name**: (*Obrigatório*) O nome do recurso personalizado de agendamento.
- **spec.appVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault da aplicação de origem.
- **spec.applicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do CR da aplicação de origem.
- **spec.backupRetention**: (*Obrigatório*) Este campo é obrigatório e o valor deve ser definido como 0.
- **spec.enabled**: deve ser definido como verdadeiro.
- **spec.granularidade**: Deve ser definida como `Custom`.
- **spec.recurrenceRule**: Defina uma data de início em UTC e um intervalo de recorrência.
- **spec.snapshotRetention**: Deve ser definido como 2.

Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: Schedule
metadata:
  name: appmirror-schedule
  namespace: my-app-namespace
spec:
  appVaultRef: my-appvault-name
  applicationRef: my-app-name
  backupRetention: "0"
  enabled: true
  granularity: Custom
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  snapshotRetention: "2"
```

i. Após preencher o arquivo `trident-protect-schedule.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-schedule.yaml -n my-app-namespace
```

Crie o agendamento de replicação usando a CLI

a. Crie o cronograma de replicação, substituindo os valores entre colchetes pelas informações do seu ambiente:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule <rule> --snapshot-retention
<snapshot_retention_count> -n <my_app_namespace>
```

Exemplo:

```
tridentctl-protect create schedule --name appmirror-schedule
--app <my_app_name> --appvault <my_app_vault> --granularity
Custom --recurrence-rule  "DTSTART:20220101T000200Z
\nRRULE:FREQ=MINUTELY;INTERVAL=5" --snapshot-retention 2 -n
<my_app_namespace>
```

5. No cluster de destino, crie um AppVault CR de aplicativo de origem idêntico ao AppVault CR que você aplicou no cluster de origem e nomeie-o (por exemplo, `trident-protect-appvault-primary-destination.yaml`).
6. Aplique o CR:

```
kubectl apply -f trident-protect-appvault-primary-destination.yaml -n
trident-protect
```

7. Crie um AppVault CR de destino para o aplicativo de destino no cluster de destino. Dependendo do seu provedor de armazenamento, modifique um exemplo em "Recursos personalizados do AppVault" para adequá-lo ao seu ambiente:
   a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-appvault-secondary-destination.yaml`).
   b. Configurar os seguintes atributos:
      - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado AppVault. Anote o nome escolhido, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esse valor.
      - **spec.providerConfig**: (*Obrigatório*) Armazena a configuração necessária para acessar o AppVault usando o provedor especificado. Escolha um `bucketName` e quaisquer outros detalhes necessários para o seu provedor. Anote os valores escolhidos, pois outros arquivos CR necessários para um relacionamento de replicação fazem referência a esses valores. Consulte "Recursos personalizados do AppVault" para exemplos de AppVault CRs com outros provedores.
      - **spec.providerCredentials**: (*Obrigatório*) Armazena referências a quaisquer credencial necessárias para acessar o AppVault usando o provedor especificado.
        - **spec.providerCredentials.valueFromSecret**: (*Obrigatório*) Indica que o valor da credencial

deve vir de um segredo.

- **chave**: (*Obrigatório*) A chave válida do segredo a ser selecionada.
- **nome**: (*Obrigatório*) Nome do segredo que contém o valor deste campo. Deve estar no mesmo namespace.

- **spec.providerCredentials.secretAccessKey**: (*Obrigatório*) A chave de acesso usada para acessar o provedor. O **name** deve corresponder a **spec.providerCredentials.valueFromSecret.name**.

- **spec.providerType**: (*Obrigatório*) Determina o que fornece o backup; por exemplo, NetApp ONTAP S3, S3 genérico, Google Cloud ou Microsoft Azure. Valores possíveis:
  - aws
  - azure
  - gcp
  - generic-s3
  - ontap-s3
  - storagegrid-s3

c. Após preencher o arquivo `trident-protect-appvault-secondary-destination.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-appvault-secondary-destination.yaml
-n trident-protect
```

8. No cluster de destino, crie um arquivo CR AppMirrorRelationship.

> Ao usar uma CR, crie manualmente o namespace de destino antes de aplicar a CR. Trident Protect cria namespaces automaticamente apenas quando se usa a CLI.

Crie um AppMirrorRelationship usando um CR

a. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-relationship.yaml`).

b. Configurar os seguintes atributos:

- **metadata.name:** (Obrigatório) O nome do recurso personalizado AppMirrorRelationship.
- **spec.destinationAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de destino no cluster de destino.
- **spec.namespaceMapping**: (*Obrigatório*) Os namespaces de destino e de origem devem corresponder ao namespace do aplicativo definido no respectivo CR do aplicativo.
- **spec.sourceAppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao nome do AppVault para o aplicativo de origem.
- **spec.sourceApplicationName**: (*Obrigatório*) Este valor deve corresponder ao nome do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.sourceApplicationUID**: (Obrigatório) Este valor deve corresponder ao UID do aplicativo de origem que você definiu no CR do aplicativo de origem.
- **spec.storageClassName**: (*Opcional*) Escolha o nome de uma classe de armazenamento válida no cluster. A classe de armazenamento deve estar vinculada a uma VM de armazenamento ONTAP que esteja emparelhada com o ambiente de origem. Se a classe de armazenamento não for fornecida, a classe de armazenamento padrão do cluster será usada por padrão.
- **spec.recurrenceRule**: Defina uma data de início em UTC e um intervalo de recorrência.

  Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: AppMirrorRelationship
metadata:
  name: amr-16061e80-1b05-4e80-9d26-d326dc1953d8
  namespace: my-app-namespace
spec:
  desiredState: Established
  destinationAppVaultRef: generic-s3-trident-protect-dst-bucket-
8fe0b902-f369-4317-93d1-ad7f2edc02b5
  namespaceMapping:
    - destination: my-app-namespace
      source: my-app-namespace
  recurrenceRule: |-
    DTSTART:20220101T000200Z
    RRULE:FREQ=MINUTELY;INTERVAL=5
  sourceAppVaultRef: generic-s3-trident-protect-src-bucket-
b643cc50-0429-4ad5-971f-ac4a83621922
  sourceApplicationName: my-app-name
  sourceApplicationUID: 7498d32c-328e-4ddd-9029-122540866aeb
  storageClassName: sc-vsim-2
```

c. Após preencher o arquivo `trident-protect-relationship.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-
namespace
```

Crie um AppMirrorRelationship usando a CLI

a. Crie e aplique o objeto AppMirrorRelationship, substituindo os valores entre colchetes por informações do seu ambiente:

```
tridentctl-protect create appmirrorrelationship
<name_of_appmirorrelationship> --destination-app-vault
<my_vault_name> --source-app-vault <my_vault_name> --recurrence
-rule <rule> --namespace-mapping <ns_mapping> --source-app-id
<source_app_UID> --source-app <my_source_app_name> --storage
-class <storage_class_name> -n <application_namespace>
```

Exemplo:

```
tridentctl-protect create appmirrorrelationship my-amr
--destination-app-vault appvault2 --source-app-vault appvault1
--recurrence-rule
"DTSTART:20220101T000200Z\nRRULE:FREQ=MINUTELY;INTERVAL=5"
--source-app my-app --namespace-mapping "my-source-ns1:my-dest-
ns1,my-source-ns2:my-dest-ns2" --source-app-id 373f24c1-5769-
404c-93c3-5538af6ccc36 --storage-class my-storage-class -n my-
dest-ns1
```

9. (*Opcional*) No cluster de destino, verifique o estado e o status da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Fail over para o cluster de destino

Com Trident Protect, você pode realizar o failover de aplicativos replicados para um cluster de destino. Este procedimento interrompe a relação de replicação e coloca o app online no cluster de destino. Trident Protect não interrompe o app no cluster de origem se ele estiver operacional.

Passos

1. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de **spec.desiredState** para `Promoted`.
2. Salve o arquivo CR.
3. Aplique o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

4. (*Opcional*) Crie quaisquer cronogramas de proteção que você precise no aplicativo com failover.
5. (*Opcional*) Verifique o estado e o status da relação de replicação:

```
kubectl get amr -n my-app-namespace <relationship name> -o=jsonpath
='{.status}' | jq
```

Ressincronizar uma relação de replicação com failover

A operação de ressincronização restabelece a relação de replicação. Após a execução de uma operação de ressincronização, o aplicativo de origem original torna-se o aplicativo em execução e quaisquer alterações feitas no aplicativo em execução no cluster de destino são descartadas.

O processo interrompe o app no cluster de destino antes de restabelecer a replicação.

Quaisquer dados gravados no aplicativo de destino durante a transição de falha serão perdidos.

Passos

1. Opcional: No cluster de origem, crie um snapshot do aplicativo de origem. Isso garante que as alterações mais recentes do cluster de origem sejam capturadas.
2. No cluster de destino, edite o arquivo CR AppMirrorRelationship (por exemplo, `trident-protect-relationship.yaml`) e altere o valor de spec.desiredState para `Established`.
3. Salve o arquivo CR.
4. Aplique o CR:

```
kubectl apply -f trident-protect-relationship.yaml -n my-app-namespace
```

5. Se você criou algum agendamento de proteção no cluster de destino para proteger o aplicativo em failover, remova-os. Quaisquer agendamentos restantes causam falhas no snapshot de volume.

Resincronizar reversamente uma relação de replicação com failover

Ao reverter a resincronização de uma relação de replicação com falha, o aplicativo de destino torna-se o aplicativo de origem e a origem torna-se o destino. As alterações feitas no aplicativo de destino durante a falha são mantidas.

Passos

1. No cluster de destino original, exclua o CR AppMirrorRelationship. Isso faz com que o destino se torne o cluster de origem. Se houver algum agendamento de proteção restante no novo cluster de destino, remova-os.
2. Estabeleça uma relação de replicação aplicando os arquivos CR que você usou originalmente para configurar a relação aos clusters opostos.
3. Certifique-se de que o novo destino (cluster de origem) esteja configurado com ambas as CRs AppVault.
4. Configure uma relação de replicação no cluster oposto, configurando valores para a direção inversa.

Inverter a direção da replicação do aplicativo

Ao inverter a direção da replicação, Trident Protect move a aplicação para o backend de armazenamento de destino, enquanto continua a replicar de volta para o backend de armazenamento de origem original. Trident Protect interrompe a aplicação de origem e replica os dados para o destino antes de realizar o failover para a aplicação de destino.

Nessa situação, você está trocando o cluster de origem e o cluster de destino.

Passos

1. No cluster de origem, crie um snapshot de desligamento:

Crie um instantâneo de desligamento usando uma CR

a. Desative os agendamentos da política de proteção para o aplicativo de origem.

b. Crie um arquivo CR ShutdownSnapshot:

   i. Crie o arquivo de recurso personalizado (CR) e dê um nome a ele (por exemplo, `trident-protect-shutdownsnapshot.yaml`).

   ii. Configurar os seguintes atributos:

   - **metadata.name**: (*Obrigatório*) O nome do recurso personalizado.
   - **spec.AppVaultRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do AppVault da aplicação de origem.
   - **spec.ApplicationRef**: (*Obrigatório*) Este valor deve corresponder ao campo metadata.name do arquivo CR do aplicativo de origem.

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ShutdownSnapshot
metadata:
  name: replication-shutdown-snapshot-afc4c564-e700-4b72-86c3-
c08a5dbe844e
  namespace: my-app-namespace
spec:
  appVaultRef: generic-s3-trident-protect-src-bucket-04b6b4ec-
46a3-420a-b351-45795e1b5e34
  applicationRef: my-app-name
```

c. Após preencher o arquivo `trident-protect-shutdownsnapshot.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-shutdownsnapshot.yaml -n my-app-
namespace
```

Crie um Snapshot de desligamento usando a CLI

a. Crie o snapshot de desligamento, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

```
tridentctl-protect create shutdownsnapshot <my_shutdown_snapshot>
--appvault <my_vault> --app <app_to_snapshot> -n
<application_namespace>
```

2. No cluster de origem, após a conclusão do snapshot de desligamento, obtenha o status do snapshot de desligamento:

```
kubectl get shutdownsnapshot -n my-app-namespace
<shutdown_snapshot_name> -o yaml
```

3. No cluster de origem, encontre o valor de **shutdownsnapshot.status.appArchivePath** usando o seguinte comando e registre a última parte do caminho do arquivo (também chamada de nome base; isso será tudo o que vem depois da última barra):

```
k get shutdownsnapshot -n my-app-namespace <shutdown_snapshot_name> -o
jsonpath='{.status.appArchivePath}'
```

4. Realize um failover do novo cluster de destino para o novo cluster de origem, com a seguinte alteração:

> Na etapa 2 do procedimento de failover, inclua o campo `spec.promotedSnapshot` no arquivo CR AppMirrorRelationship e defina seu valor para o basename que você registrou na etapa 3 acima.

5. Execute as etapas de resincronização reversa em Resincronizar reversamente uma relação de replicação com failover.
6. Ative os agendamentos de proteção no novo cluster de origem.

Resultado

As seguintes ações ocorrem devido à replicação reversa:

- É feita uma snapshot dos recursos Kubernetes do aplicativo de origem original.
- Os pods do aplicativo de origem original são interrompidos corretamente ao excluir os recursos do Kubernetes do aplicativo (mantendo os PVCs e PVs no lugar).
- Após o desligamento dos pods, snapshots dos volumes do aplicativo são criados e replicados.
- As relações do SnapMirror são rompidas, tornando os volumes de destino prontos para leitura/gravação.
- Os recursos do Kubernetes do aplicativo são restaurados a partir do Snapshot anterior ao desligamento, usando os dados de volume replicados após o desligamento do aplicativo de origem original.
- A replicação é restabelecida na direção inversa.

Restaurar aplicativos para o cluster de origem

Com Trident Protect, você pode realizar o "fail back" após uma operação de failover seguindo a seguinte sequência de operações. Nesse fluxo de trabalho para restaurar a direção de replicação original, Trident Protect replica (ressincroniza) quaisquer alterações do aplicativo de volta para o aplicativo de origem original antes de reverter a direção da replicação.

Esse processo começa a partir de uma relação que concluiu um failover para um destino e envolve as seguintes etapas:

- Comece com um estado de failover.

- Reverter a resincronização da relação de replicação.

> Não execute uma operação de ressincronização normal, pois isso descartará dados gravados no cluster de destino durante o procedimento de fail over.

- Inverta a direção da replicação.

Passos

1. Execute as etapas Resincronizar reversamente uma relação de replicação com failover.
2. Execute os Inverter a direção da replicação do aplicativo passos.

Excluir uma relação de replicação

Você pode excluir uma relação de replicação a qualquer momento. Ao excluir a relação de replicação do aplicativo, o resultado são dois aplicativos separados, sem nenhuma relação entre eles.

Passos

1. No cluster de destino atual, exclua o AppMirrorRelationship CR:

```
kubectl delete -f trident-protect-relationship.yaml -n my-app-namespace
```

Migrar aplicativos usando Trident Protect

Você pode migrar seus aplicativos entre clusters ou para diferentes classes de armazenamento restaurando dados de backup.

> Ao migrar um aplicativo, todos os ganchos de execução configurados para o aplicativo são migrados juntamente com o aplicativo. Se houver um gancho de execução pós-restauração, ele é executado automaticamente como parte da operação de restauração.

Operações de backup e restauração

Para executar operações de backup e restauração nos seguintes cenários, você pode automatizar tarefas específicas de backup e restauração.

Clonar para o mesmo cluster

Para clonar um aplicativo para o mesmo cluster, crie um snapshot ou backup e restaure os dados para o mesmo cluster.

Passos

1. Faça uma das seguintes ações:
   a. "Criar um snapshot".
   b. "Criar um backup".
2. No mesmo cluster, faça um dos seguintes procedimentos, dependendo se você criou um snapshot ou um backup:
   a. "Restaure seus dados a partir do snapshot".

b. "Restaure seus dados do backup".

Clonar para cluster diferente

Para clonar um aplicativo para um cluster diferente (realizar uma clonagem entre clusters), crie um backup no cluster de origem e, em seguida, restaure o backup em um cluster diferente. Certifique-se de que Trident Protect esteja instalado no cluster de destino.

Você pode replicar um aplicativo entre diferentes clusters usando "SnapMirror replicação".

Passos

1. "Criar um backup".
2. Certifique-se de que o CR AppVault para o bucket do storage de objetos que contém o backup foi configurado no cluster de destino.
3. No cluster de destino, "restaure seus dados a partir do backup".

Migrar aplicações de uma classe de armazenamento para outra classe de armazenamento

Você pode migrar aplicativos de uma classe de armazenamento para outra restaurando um backup na classe de armazenamento de destino.

Por exemplo (excluindo os segredos do CR de restauração):

```
apiVersion: protect.trident.netapp.io/v1
kind: SnapshotRestore
metadata:
  name: "${snapshotRestoreCRName}"
spec:
  appArchivePath: "${snapshotArchivePath}"
  appVaultRef: "${appVaultCRName}"
  namespaceMapping:
    - destination: "${destinationNamespace}"
      source: "${sourceNamespace}"
  storageClassMapping:
    - destination: "${destinationStorageClass}"
      source: "${sourceStorageClass}"
  resourceFilter:
    resourceMatchers:
      kind: Secret
      version: v1
    resourceSelectionCriteria: exclude
```

Restaure o snapshot usando um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-snapshot-restore-cr.yaml`.
2. No arquivo que você criou, configure os seguintes atributos:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do snapshot está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get snapshots <my-snapshot-name> -n trident-protect -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do snapshot está armazenado.
   - **spec.namespaceMapping**: O mapeamento do namespace de origem da operação de restauração para o namespace de destino. Substitua `my-source-namespace` e `my-destination-namespace` pelas informações do seu ambiente.

     ```
     ---
     apiVersion: protect.trident.netapp.io/v1
     kind: SnapshotRestore
     metadata:
       name: my-cr-name
       namespace: trident-protect
     spec:
       appArchivePath: my-snapshot-path
       appVaultRef: appvault-name
       namespaceMapping: [{"source": "my-source-namespace",
     "destination": "my-destination-namespace"}]
     ```

3. Opcionalmente, se precisar selecionar apenas determinados recursos do aplicativo para restaurar, adicione filtragem que inclua ou exclua recursos marcados com rótulos específicos:
   - **resourceFilter.resourceSelectionCriteria**: (Obrigatório para filtragem) Use `include or exclude` para incluir ou excluir um recurso definido em resourceMatchers. Adicione os seguintes parâmetros resourceMatchers para definir os recursos a serem incluídos ou excluídos:
     - **resourceFilter.resourceMatchers**: Uma matriz de objetos resourceMatcher. Se você definir vários elementos nesta matriz, eles correspondem como uma operação OR, e os campos dentro de cada elemento (group, kind, version) correspondem como uma operação AND.
       - **resourceMatchers[].group**: (*Opcional*) Grupo do recurso a ser filtrado.
       - **resourceMatchers[].kind**: (*Opcional*) Tipo do recurso a ser filtrado.
       - **resourceMatchers[].version**: (Opcional) Versão do recurso a ser filtrado.

- **resourceMatchers[].names**: (*Opcional*) Nomes no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].namespaces**: (*Opcional*) Namespaces no campo metadata.name do Kubernetes do recurso a ser filtrado.
- **resourceMatchers[].labelSelectors**: (*Opcional*) String seletora de rótulo no campo metadata.name do Kubernetes do recurso, conforme definido no "Documentação do Kubernetes". Por exemplo: `"trident.netapp.io/os=linux"`.

  Por exemplo:

```
spec:
  resourceFilter:
    resourceSelectionCriteria: "include"
    resourceMatchers:
      - group: my-resource-group-1
        kind: my-resource-kind-1
        version: my-resource-version-1
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
      - group: my-resource-group-2
        kind: my-resource-kind-2
        version: my-resource-version-2
        names: ["my-resource-names"]
        namespaces: ["my-resource-namespaces"]
        labelSelectors: ["trident.netapp.io/os=linux"]
```

4. Após preencher o arquivo `trident-protect-snapshot-restore-cr.yaml` com os valores corretos, aplique a CR:

```
kubectl apply -f trident-protect-snapshot-restore-cr.yaml
```

Restaurar o snapshot usando a CLI

Passos

1. Restaure o snapshot para um namespace diferente, substituindo os valores entre colchetes pelas informações do seu ambiente.

   - O `snapshot` argumento usa um namespace e um nome de snapshot no formato `<namespace>/<name>`.
   - O `namespace-mapping`argumento usa namespaces separados por dois pontos para mapear namespaces de origem para os namespaces de destino corretos no formato `source1:dest1,source2:dest2`.

     Por exemplo:

```
tridentctl-protect create snapshotrestore <my_restore_name>
--snapshot <namespace/snapshot_to_restore> --namespace-mapping
<source_to_destination_namespace_mapping>
```

Gerenciar os ganchos de execução do Trident Protect

Um gancho de execução é uma ação personalizada que você pode configurar para ser executada em conjunto com uma operação de proteção de dados de um aplicativo gerenciado. Por exemplo, se você tiver um aplicativo de banco de dados, pode usar um gancho de execução para pausar todas as transações do banco de dados antes de um Snapshot e retomá-las após a conclusão do Snapshot. Isso garante Snapshots consistentes para o aplicativo.

Tipos de ganchos de execução

Trident Protect suporta os seguintes tipos de ganchos de execução, com base em quando podem ser executados:

- Pré-Snapshot
- Pós-Snapshot
- Pré-backup
- Pós-backup
- Pós-restauração
- Pós-failover

Ordem de execução

Quando uma operação de proteção de dados é executada, os eventos de gancho de execução ocorrem na seguinte ordem:

1. Quaisquer hooks de execução pré-operacionais personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos hooks de pré-operação personalizados forem necessários, mas a ordem de execução desses hooks antes da operação não é garantida nem configurável.
2. Ocorrem congelamentos do sistema de arquivos, se aplicável. "Saiba mais sobre como configurar o congelamento do sistema de arquivos com Trident Protect".
3. A operação de proteção de dados é realizada.
4. Os sistemas de arquivos congelados são descongelados, se aplicável.
5. Quaisquer hooks de execução pós-operação personalizados aplicáveis são executados nos contêineres apropriados. Você pode criar e executar quantos hooks de pós-operação personalizados forem necessários, mas a ordem de execução desses hooks após a operação não é garantida nem configurável.

Se você criar vários ganchos de execução do mesmo tipo (por exemplo, pré-snapshot), a ordem de execução desses ganchos não é garantida. No entanto, a ordem de execução de ganchos de tipos diferentes é garantida. Por exemplo, a seguir está a ordem de execução de uma configuração que possui todos os

diferentes tipos de ganchos:

1. Ganchos pré-Snapshot executados
2. Ganchos pós-Snapshot executados
3. Ganchos de pré-backup executados
4. Ganchos pós-backup executados

O exemplo de ordem anterior só se aplica quando você executa um backup que não usa um snapshot existente.

Você deve sempre testar seus scripts de gancho de execução antes de habilitá-los em um ambiente de produção. Você pode usar o comando 'kubectl exec' para testar os scripts convenientemente. Depois de habilitar os ganchos de execução em um ambiente de produção, teste os snapshots e backups resultantes para garantir que estejam consistentes. Você pode fazer isso clonando o app para um namespace temporário, restaurando o snapshot ou backup e, em seguida, testando o app.

Se um gancho de execução pré-snapshot adicionar, alterar ou remover recursos do Kubernetes, essas alterações serão incluídas no snapshot ou backup e em qualquer operação de restauração subsequente.

Observações importantes sobre ganchos de execução personalizados

Considere o seguinte ao planejar os ganchos de execução para seus aplicativos.

- Um gancho de execução deve usar um script para executar ações. Vários ganchos de execução podem referenciar o mesmo script.
- Trident Protect exige que os scripts que os ganchos de execução usam sejam escritos no formato de scripts shell executáveis.
- O tamanho do script está limitado a 96KB.
- Trident Protect utiliza as configurações de ganchos de execução e quaisquer critérios correspondentes para determinar quais ganchos são aplicáveis a uma operação de snapshot, backup ou restauração.

Como os hooks de execução frequentemente reduzem ou desativam completamente a funcionalidade do aplicativo em que estão sendo executados, você deve sempre tentar minimizar o tempo que seus hooks de execução personalizados levam para serem executados. Se você iniciar uma operação de backup ou snapshot com hooks de execução associados, mas depois cancelá-la, os hooks ainda poderão ser executados se a operação de backup ou snapshot já tiver sido iniciada. Isso significa que a lógica usada em um hook de execução pós-backup não pode presumir que o backup foi concluído.

Filtros de gancho de execução

Ao adicionar ou editar um gancho de execução para um aplicativo, você pode adicionar filtros ao gancho de execução para gerenciar quais contêineres o gancho irá corresponder. Os filtros são úteis para aplicativos que usam a mesma imagem de contêiner em todos os contêineres, mas podem usar cada imagem para uma finalidade diferente (como Elasticsearch). Os filtros permitem criar cenários em que ganchos de execução são executados em alguns, mas não necessariamente em todos os contêineres idênticos. Se você criar vários filtros para um único gancho de execução, eles serão combinados com um operador lógico AND. Você pode ter até 10 filtros ativos por gancho de execução.

Cada filtro que você adiciona a um gancho de execução usa uma expressão regular para corresponder contêineres no seu cluster. Quando um gancho corresponde a um contêiner, o gancho executará seu script associado nesse contêiner. As expressões regulares para filtros usam a sintaxe Regular Expression 2 (RE2), que não permite criar um filtro que exclua contêineres da lista de correspondências. Para obter informações sobre a sintaxe que o Trident Protect oferece suporte para expressões regulares em filtros de ganchos de execução, consulte "Suporte à sintaxe de Regular Expression 2 (RE2)".

> Se você adicionar um filtro de namespace a um gancho de execução que é executado após uma operação de restauração ou clonagem e a origem e o destino estiverem em namespaces diferentes, o filtro de namespace será aplicado apenas ao namespace de destino.

Exemplos de ganchos de execução

Visite o "Projeto NetApp Verda GitHub" para baixar hooks de execução reais para aplicativos populares como Apache Cassandra e Elasticsearch. Você também pode ver exemplos e obter ideias para estruturar seus próprios hooks de execução personalizados.

Crie um gancho de execução

Você pode criar um gancho de execução personalizado para um aplicativo usando Trident Protect. Você precisa ter permissões de Proprietário, Administrador ou Membro para criar ganchos de execução.

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do aplicativo Kubernetes para o qual o gancho de execução será executado.
   - **spec.stage**: (*Obrigatório*) Uma string indicando em qual etapa da ação o gancho de execução deve ser executado. Valores possíveis:
     - Pré
     - Publicar
   - **spec.action**: (*Obrigatório*) Uma string indicando qual ação o gancho de execução executará, assumindo que quaisquer filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **spec.enabled**: (opcional) Indica se este gancho de execução está habilitado ou desabilitado. Se não for especificado, o valor padrão é true.
   - **spec.hookSource**: (*Obrigatório*) Uma string contendo o script de gancho codificado em base64.
   - **spec.timeout**: (Opcional) Um número que define por quanto tempo, em minutos, o gancho de execução pode ser executado. O valor mínimo é 1 minuto e o valor padrão é 25 minutos se não for especificado.
   - **spec.arguments**: (opcional) Uma lista YAML de argumentos que você pode especificar para o gancho de execução.
   - **spec.matchingCriteria**: (*Opcional*) Uma lista opcional de pares de chave-valor de critérios, cada par constituindo um filtro de gancho de execução. Você pode adicionar até 10 filtros por gancho de execução.
   - **spec.matchingCriteria.type**: (Opcional) Uma string que identifica o tipo de filtro do gancho de execução. Valores possíveis:
     - ContainerImage
     - ContainerName
     - PodName
     - PodLabel
     - NamespaceName
   - **spec.matchingCriteria.value**: (*Opcional*) Uma string ou expressão regular que identifica o valor do filtro do gancho de execução.

     Exemplo YAML:

```
apiVersion: protect.trident.netapp.io/v1
kind: ExecHook
metadata:
  name: example-hook-cr
  namespace: my-app-namespace
  annotations:
    astra.netapp.io/astra-control-hook-source-id:
/account/test/hookSource/id
spec:
  applicationRef: my-app-name
  stage: Pre
  action: Snapshot
  enabled: true
  hookSource: IyEvYmluL2Jhc2gKZWNobyAiZXhhbXBsZSBzY3JpcHQiCg==
  timeout: 10
  arguments:
    - FirstExampleArg
    - SecondExampleArg
  matchingCriteria:
    - type: containerName
      value: mysql
    - type: containerImage
      value: bitnami/mysql
    - type: podName
      value: mysql
    - type: namespaceName
      value: mysql-a
    - type: podLabel
      value: app.kubernetes.io/component=primary
    - type: podLabel
      value: helm.sh/chart=mysql-10.1.0
    - type: podLabel
      value: deployment-type=production
```

3. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook.yaml
```

Use o CLI

Passos

1. Crie o gancho de execução, substituindo os valores entre colchetes por informações do seu ambiente. Por exemplo:

```
tridentctl-protect create exechook <my_exec_hook_name> --action
<action_type> --app <app_to_use_hook> --stage <pre_or_post_stage>
--source-file <script-file> -n <application_namespace>
```

Executar manualmente um gancho de execução

Você pode executar um gancho de execução manualmente para fins de teste ou se precisar executá-lo novamente após uma falha. Você precisa ter permissões de Proprietário, Administrador ou Membro para executar ganchos de execução manualmente.

A execução manual de um gancho de execução consiste em duas etapas básicas:

1. Crie um backup de recurso, que coleta recursos e cria um backup deles, determinando onde o gancho será executado
2. Execute o hook de execução no backup

Passo 1: Crie um backup de recurso

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-resource-backup.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) O nome do Kubernetes da aplicação para a qual criar o backup do recurso.
   - **spec.appVaultRef**: (*Obrigatório*) O nome do AppVault onde o conteúdo do backup está armazenado.
   - **spec.appArchivePath**: O caminho dentro do AppVault onde o conteúdo do backup está armazenado. Você pode usar o seguinte comando para encontrar esse caminho:

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ResourceBackup
metadata:
  name: example-resource-backup
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
```

3. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-resource-backup.yaml
```

Use o CLI

Passos

1. Crie o backup, substituindo os valores entre colchetes pelas informações do seu ambiente. Por exemplo:

```
tridentctl protect create resourcebackup <my_backup_name> --app
<my_app_name> --appvault <my_appvault_name> -n
<my_app_namespace> --app-archive-path <app_archive_path>
```

2. Veja o status do backup. Você pode usar este comando de exemplo repetidamente até que a operação seja concluída:

```
tridentctl protect get resourcebackup -n <my_app_namespace>
<my_backup_name>
```

3. Verifique se o backup foi bem-sucedido:

```
kubectl describe resourcebackup <my_backup_name>
```

Etapa 2: execute o gancho de execução

Use um CR

Passos

1. Crie o arquivo de recurso personalizado (CR) e nomeie-o `trident-protect-hook-run.yaml`.
2. Configure os seguintes atributos para corresponder ao seu ambiente Trident Protect e à configuração do cluster:
   - **metadata.name**: (*Obrigatório*) O nome deste recurso personalizado; escolha um nome único e adequado ao seu ambiente.
   - **spec.applicationRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao nome do aplicativo da ResourceBackup CR que você criou na etapa 1.
   - **spec.appVaultRef**: (*Obrigatório*) Certifique-se de que este valor corresponda ao appVaultRef do ResourceBackup CR que você criou na etapa 1.
   - **spec.appArchivePath**: Certifique-se de que este valor corresponda ao appArchivePath do ResourceBackup CR que você criou na etapa 1.

     ```
     kubectl get backups <BACKUP_NAME> -n my-app-namespace -o jsonpath='{.status.appArchivePath}'
     ```

   - **spec.action**: (*Obrigatório*) Uma string indicando qual ação o gancho de execução executará, assumindo que quaisquer filtros de gancho de execução especificados sejam correspondidos. Valores possíveis:
     - Snapshot
     - Backup
     - Restaurar
     - Failover
   - **spec.stage**: (*Obrigatório*) Uma string indicando em qual estágio da ação o gancho de execução deve ser executado. Essa execução de gancho não executará ganchos em nenhum outro estágio. Valores possíveis:
     - Pré
     - Publicar

     Exemplo YAML:

```
---
apiVersion: protect.trident.netapp.io/v1
kind: ExecHooksRun
metadata:
  name: example-hook-run
spec:
  applicationRef: my-app-name
  appVaultRef: my-appvault-name
  appArchivePath: example-resource-backup
  stage: Post
  action: Failover
```

3. Após preencher o arquivo CR com os valores corretos, aplique o CR:

```
kubectl apply -f trident-protect-hook-run.yaml
```

Use o CLI

Passos

1. Crie a solicitação de execução manual do hook:

```
tridentctl protect create exechooksrun <my_exec_hook_run_name>
-n <my_app_namespace> --action snapshot --stage <pre_or_post>
--app <my_app_name> --appvault <my_appvault_name> --path
<my_backup_name>
```

2. Verifique o status da execução do gancho. Você pode executar este comando repetidamente até que a operação seja concluída:

```
tridentctl protect get exechooksrun -n <my_app_namespace>
<my_exec_hook_run_name>
```

3. Descreva o objeto exechooksrun para ver os detalhes finais e o status:

```
kubectl -n <my_app_namespace> describe exechooksrun
<my_exec_hook_run_name>
```

Desinstale Trident Protect

Você pode precisar remover componentes do Trident Protect se estiver atualizando de uma versão de avaliação para uma versão completa do produto.

Para remover Trident Protect, execute as seguintes etapas.

Passos

1. Remova os arquivos CR do Trident Protect:

   Esta etapa não é necessária para a versão 25.06 e posteriores.

   ```
   helm uninstall -n trident-protect trident-protect-crds
   ```

2. Remover Trident Protect:

   ```
   helm uninstall -n trident-protect trident-protect
   ```

3. Remova o namespace Trident Protect:

   ```
   kubectl delete ns trident-protect
   ```

Blogs do Trident e Trident Protect

Você pode encontrar alguns ótimos blogs sobre NetApp Trident e Trident Protect aqui:

Blogs do Trident

- 16 de outubro de 2025: "Soluções avançadas de storage para Kubernetes"
- 19 de agosto de 2025: "Aprimorando a consistência de dados: snapshots de grupos de volumes em OpenShift virtualização com Trident"
- 09 de maio de 2025: "Configuração automática do backend Trident para FSx para ONTAP com o add-on Amazon EKS"
- 15 de abril de 2025: "NetApp Trident com Google Cloud NetApp Volumes para protocolo SMB"
- 14 de abril de 2025: "Aproveitando o protocolo Fiber Channel com Trident 25.02 para armazenamento persistente no Kubernetes"
- 14 de abril de 2025: "Desbloqueando o poder dos sistemas NetApp ASA r2 para armazenamento em blocos do Kubernetes"
- 31 de março de 2025: "Simplificando a instalação do Trident no Red Hat OpenShift com o novo operador certificado"
- 27 de março de 2025: "Provisionando Trident para SMB com Google Cloud NetApp Volumes"
- 05 de março de 2025: "Desbloqueie a integração perfeita de armazenamento iSCSI: um guia para FSxN em clusters ROSA para AWS"
- 27 de fevereiro de 2025: "Implantação de identidade na nuvem com Trident, GKE e Google Cloud NetApp Volumes"
- 12 de dezembro de 2024: "Apresentando suporte a Fibre Channel no Trident"
- 11 de novembro de 2024: "NetApp Trident com Google Cloud NetApp Volumes"
- 29 de outubro de 2024: "Amazon FSx para NetApp ONTAP com Red Hat OpenShift Service on AWS (ROSA) usando Trident"
- 29 de outubro de 2024: "Migração em tempo real de VMs com OpenShift Virtualization no ROSA e Amazon FSx for NetApp ONTAP"
- 08 de julho de 2024: "Utilizando NVMe/TCP para consumir armazenamento ONTAP para seus aplicativos modernos em contêineres no Amazon EKS"
- 01 de julho de 2024: "Armazenamento Kubernetes integrado com Google Cloud NetApp Volumes Flex e Astra Trident"
- 11 de junho de 2024: "ONTAP como armazenamento de backend para o registro de imagens integrado em OpenShift"

Blogs do Trident Protect

- 16 de maio de 2025: "Automatizando o failover do registro para recuperação de desastres com os hooks pós-restauração do Trident Protect"
- 16 de maio de 2025: "OpenShift Virtualização Recuperação de desastres com NetApp Trident Protect"
- 13 de maio de 2025: "Migração de storage class com Trident Protect backup restore"
- 09 de maio de 2025: "Redimensione aplicações Kubernetes com os hooks de pós-restauração do Trident

Protect"

- 03 de abril de 2025: "Trident Protect Power Up: replicação do Kubernetes para proteção e recuperação de desastres"
- 13 de março de 2025: "Operações de backup e restauração consistentes em caso de falha para OpenShift Virtualization VMs"
- 11 de março de 2025: "Estendendo os padrões de GitOps à proteção de dados de aplicativos com NetApp Trident"
- 03 de mar de 2025: "Trident 25.02: Elevando a experiência Red Hat OpenShift com novos e empolgantes recursos"
- 15 de janeiro de 2025: "Apresentando o controle de acesso baseado em funções do Trident Protect"
- 11 de novembro de 2024: "Gerenciamento de dados orientado por Kubernetes: a nova era com Trident Protect"

Conhecimento e suporte

Perguntas frequentes

Encontre respostas para as perguntas mais frequentes sobre instalação, configuração, atualização e resolução de problemas do Trident.

Perguntas gerais

Com que frequência o Trident é lançado?

A partir da versão 24.02, Trident é lançado a cada quatro meses: fevereiro, junho e outubro.

O Trident é compatível com todos os recursos lançados em uma determinada versão do Kubernetes?

Trident geralmente não oferece suporte a recursos alfa no Kubernetes. Trident pode oferecer suporte a recursos beta nas duas versões do Trident subsequentes à versão beta do Kubernetes.

O Trident tem alguma dependência de outros produtos NetApp para seu funcionamento?

Trident não possui dependências de outros NetApp software products e funciona como um aplicativo independente. No entanto, você deve ter um NetApp backend storage device.

Como posso obter detalhes completos da configuração do Trident?

Utilize o `tridentctl get` comando para obter mais informações sobre a sua configuração do Trident.

Posso obter métricas sobre como o storage é provisionado pelo Trident?

Sim. Os endpoints do Prometheus podem ser usados para coletar informações sobre a operação do Trident, como o número de backends gerenciados, o número de volumes provisionados, bytes consumidos e assim por diante. Você também pode usar "Cloud Insights" para monitoramento e análise.

A experiência do usuário muda ao usar Trident como um provisionador CSI?

Não. Não há alterações em termos de experiência do usuário e funcionalidades. O nome do provisionador usado é `csi.trident.netapp.io`. Este método de instalação Trident é recomendado se você deseja usar todos os novos recursos oferecidos pelas versões atuais e futuras.

Instale e utilize Trident em um cluster Kubernetes

O Trident suporta instalação offline a partir de um registro privado?

Sim, Trident pode ser instalado offline. Consulte "Saiba mais sobre a instalação do Trident".

Posso instalar Trident remotamente?

Sim. Trident 18.10 e versões posteriores suportam a capacidade de instalação remota a partir de qualquer máquina que tenha `kubectl` acesso ao cluster. Após `kubectl` o acesso ser verificado (por exemplo, iniciar um `kubectl get nodes` comando a partir da máquina remota para verificar), siga as instruções de instalação.

Posso configurar High Availability com Trident?

Trident é instalado como um Deployment do Kubernetes (ReplicaSet) com uma instância, portanto, possui HA integrado. Você não deve aumentar o número de réplicas no deployment. Se o nó onde Trident está instalado for perdido ou o pod ficar inacessível, o Kubernetes reimplanta automaticamente o pod em um nó saudável no seu cluster. Trident é apenas plano de controle, então os pods atualmente montados não são afetados se Trident for reimplantado.

Trident precisa de acesso ao namespace kube-system?

Trident lê do Kubernetes API Server para determinar quando os aplicativos solicitam novos PVCs, então ele precisa de acesso ao kube-system.

Quais são as funções e privilégios utilizados pelo Trident?

O instalador do Trident cria um Kubernetes ClusterRole, que tem acesso específico aos PersistentVolume, PersistentVolumeClaim, StorageClass e recursos Secret do cluster Kubernetes. Consulte "Personalize a instalação do tridentctl".

Posso gerar localmente os mesmos arquivos de manifesto que o Trident usa para instalação?

Você pode gerar e modificar localmente os arquivos de manifesto exatos que o Trident usa para instalação, se necessário. Consulte "Personalize a instalação do tridentctl".

Posso compartilhar a mesma SVM de backend ONTAP para duas instâncias Trident separadas em dois clusters Kubernetes distintos?

Embora não seja recomendável, você pode usar o mesmo SVM de backend para duas instâncias do Trident. Especifique um nome de volume exclusivo para cada instância durante a instalação e/ou especifique um parâmetro exclusivo `StoragePrefix` no arquivo `setup/backend.json`. Isso é para garantir que o mesmo volume FlexVol não seja usado para ambas as instâncias.

É possível instalar Trident no ContainerLinux (antigo CoreOS)?

Trident é simplesmente um pod do Kubernetes e pode ser instalado onde o Kubernetes está em execução.

Posso usar Trident com NetApp Cloud Volumes ONTAP?

Sim, Trident é compatível com AWS, Google Cloud e Azure.

Resolução de problemas e suporte

A NetApp suporta Trident?

Embora Trident seja de código aberto e fornecido gratuitamente, NetApp oferece suporte total, desde que seu backend NetApp seja compatível.

Como faço para abrir um chamado de suporte?

Para abrir um chamado de suporte, faça um dos seguintes:

1. Entre em contato com seu Support Account Manager e obtenha ajuda para abrir um chamado.
2. Abra um chamado de suporte entrando em contato com "Suporte da NetApp".

Como faço para gerar um pacote de logs de suporte?

Você pode criar um pacote de suporte executando `tridentctl logs -a`. Além dos logs capturados no pacote, capture o log do kubelet para diagnosticar os problemas de montagem no lado do Kubernetes. As instruções para obter o log do kubelet variam dependendo de como o Kubernetes está instalado.

O que devo fazer se precisar solicitar uma nova funcionalidade?

Crie uma issue em "Trident Github" e mencione **RFE** no assunto e na descrição da issue.

Onde posso reportar um defeito?

Crie uma ocorrência em "Trident Github". Certifique-se de incluir todas as informações e registros necessários referentes ao problema.

O que acontece se eu tiver uma pergunta rápida sobre Trident que precise de esclarecimento? Existe uma comunidade ou um fórum?

Se você tiver alguma dúvida, problema ou solicitação, entre em contato conosco através do nosso Trident "Canal do Discord" ou GitHub.

A senha do meu sistema de storage foi alterada e o Trident não funciona mais, como faço para recuperar?

Atualize a senha do backend com `tridentctl update backend myBackend -f </path/to_new_backend.json> -n trident`. Substitua `myBackend` no exemplo pelo nome do seu backend, e `/path/to_new_backend.json` pelo caminho para o arquivo `backend.json` correto.

Trident não consegue encontrar meu nó do Kubernetes. Como faço para corrigir isso?

Existem dois cenários prováveis para o Trident não conseguir encontrar um nó do Kubernetes. Isso pode ser devido a um problema de rede dentro do Kubernetes ou a um problema de DNS. O daemonset do nó Trident, que é executado em cada nó do Kubernetes, deve ser capaz de se comunicar com o controlador do Trident para registrar o nó no Trident. Se alterações de rede ocorreram após a instalação do Trident, você encontrará esse problema apenas com novos nós do Kubernetes adicionados ao cluster.

Se o pod Trident for destruído, perderei os dados?

Os dados não serão perdidos se o pod do Trident for destruído. Os metadados do Trident são armazenados em objetos CRD. Todos os PVs que foram provisionados pelo Trident funcionarão normalmente.

Atualizar Trident

Posso atualizar de uma versão antiga diretamente para uma versão mais recente (pulando algumas versões)?

NetApp oferece suporte à atualização do Trident de uma versão principal para a próxima versão principal imediata. Você pode atualizar da versão 18.xx para 19.xx, de 19.xx para 20.xx e assim por diante. Você deve testar a atualização em um laboratório antes da implantação em produção.

É possível fazer o downgrade do Trident para uma versão anterior?

Se precisar corrigir erros observados após uma atualização, problemas de dependência ou uma atualização incompleta ou malsucedida, você deve "desinstalar Trident" e reinstalar a versão anterior seguindo as

instruções específicas para essa versão. Esta é a única maneira recomendada de fazer o downgrade para uma versão anterior.

Gerenciar backends e volumes

Preciso definir tanto Management quanto DataLIF em um arquivo de definição de backend ONTAP?

O gerenciamento do LIF é obrigatório. DataLIF varia:

- ONTAP SAN: não especifique para iSCSI. Trident usa "Mapa de LUN seletivo do ONTAP" para descobrir as LIFs iSCSI necessárias para estabelecer uma sessão de múltiplos caminhos. Um aviso é gerado se `dataLIF` for definido explicitamente. Consulte "Opções e exemplos de configuração do ONTAP SAN" para obter detalhes.
- ONTAP NAS: NetApp recomenda especificar `dataLIF`. Se não for fornecido, Trident busca os dataLIFs da SVM. Você pode especificar um nome de domínio totalmente qualificado (FQDN) para ser usado nas operações de montagem NFS, permitindo criar um DNS round-robin para balancear a carga entre vários dataLIFs. Consulte "Opções e exemplos de configuração do ONTAP NAS" para obter detalhes

O Trident pode configurar CHAP para backends ONTAP?

Sim. Trident suporta CHAP bidirecional para backends ONTAP. Isso requer definir `useCHAP=true` na configuração do seu backend.

Como faço para gerenciar políticas de exportação com Trident?

Trident pode criar e gerenciar políticas de exportação dinamicamente a partir da versão 20.04. Isso permite que o administrador de storage forneça um ou mais blocos CIDR em sua configuração de backend e que Trident adicione os IPs dos nós que se enquadram nesses intervalos a uma política de exportação que ele cria. Dessa forma, Trident gerencia automaticamente a adição e remoção de regras para nós com IPs dentro dos CIDRs fornecidos.

É possível usar endereços IPv6 para as Management e DataLIFs?

Trident suporta a definição de endereços IPv6 para:

- `managementLIF` e `dataLIF` para backends ONTAP NAS.
- `managementLIF` para backends SAN do ONTAP. Você não pode especificar `dataLIF` em um backend SAN do ONTAP.

Trident deve ser instalado usando a flag `--use-ipv6` (para `tridentctl` instalação), `IPv6` (para Trident operator), ou `tridentTPv6` (para instalação via Helm) para que funcione sobre IPv6.

É possível atualizar o LIF de gerenciamento no backend?

Sim, é possível atualizar o LIF de gerenciamento do backend usando o `tridentctl update backend` comando.

É possível atualizar o DataLIF no backend?

Você pode atualizar o DataLIF em `ontap-nas` e `ontap-nas-economy` apenas.

Posso criar vários backends no Trident para Kubernetes?

Trident pode suportar vários backends simultaneamente, seja com o mesmo driver ou com drivers diferentes.

Como o Trident armazena as credenciais do backend?

Trident armazena as credenciais do backend como Kubernetes Secrets.

Como o Trident seleciona um backend específico?

Caso os atributos de backend não possam ser usados para selecionar automaticamente os pools corretos para uma classe, os parâmetros `storagePools` e `additionalStoragePools` são usados para selecionar um conjunto específico de pools.

Como posso garantir que Trident não provisione a partir de um backend específico?

O `excludeStoragePools`parâmetro é usado para filtrar o conjunto de pools que Trident usa para provisionamento e removerá quaisquer pools que correspondam.

Se existirem vários backends do mesmo tipo, como o Trident seleciona qual backend usar?

Se houver vários backends configurados do mesmo tipo, Trident seleciona o backend apropriado com base nos parâmetros presentes em `StorageClass` e `PersistentVolumeClaim`. Por exemplo, se houver vários backends do driver ontap-nas, Trident tenta combinar os parâmetros em `StorageClass` e `PersistentVolumeClaim` e encontrar um backend que possa atender aos requisitos listados em `StorageClass` e `PersistentVolumeClaim`. Se houver vários backends que correspondam à solicitação, Trident seleciona um deles aleatoriamente.

Trident suporta CHAP bidirecional com Element/SolidFire?

Sim.

Como o Trident implementa Qtrees em um volume ONTAP? Quantas Qtrees podem ser implementadas em um único volume?

O `ontap-nas-economy` driver cria até 200 Qtrees no mesmo FlexVol volume (configurável entre 50 e 300), 100.000 Qtrees por nó de cluster e 2,4M por cluster. Quando você insere uma nova `PersistentVolumeClaim` que é atendida pelo driver econômico, o driver verifica se já existe um volume FlexVol que possa atender à nova Qtree. Se o volume FlexVol que possa atender à Qtree não existir, um novo volume FlexVol será criado.

Como posso configurar as permissões Unix para volumes provisionados em ONTAP NAS?

Você pode definir permissões Unix no volume provisionado pelo Trident definindo um parâmetro no arquivo de definição do backend.

Como posso configurar um conjunto explícito de opções de montagem NFS do ONTAP durante o provisionamento de um volume?

Por padrão, Trident não define opções de montagem com nenhum valor no Kubernetes. Para especificar as opções de montagem na Kubernetes Storage Class, siga o exemplo fornecido "aqui".

Como faço para definir os volumes provisionados para uma política de exportação específica?

Para permitir que os hosts apropriados acessem um volume, use o `exportPolicy` parâmetro configurado no arquivo de definição do backend.

Como faço para configurar criptografia através do Trident com ONTAP?

Você pode configurar criptografia no volume provisionado pelo Trident usando o parâmetro encryption no arquivo de definição do backend. Para obter mais informações, consulte: "Como Trident funciona com NVE e NAE"

Qual a melhor forma de implementar QoS para ONTAP através do Trident?

Use `StorageClasses` para implementar QoS para ONTAP.

Como posso especificar thin provisioning ou thick provisioning através do Trident?

Os drivers ONTAP suportam thin provisioning ou thick provisioning. Os drivers ONTAP utilizam thin provisioning por padrão. Se for desejado thick provisioning, você deve configurar o arquivo de definição de backend ou o `StorageClass`. Se ambos estiverem configurados, `StorageClass` terá precedência. Configure o seguinte para ONTAP:

1. Em `StorageClass`, defina o atributo `provisioningType` como espesso.
2. No arquivo de definição do backend, habilite volumes espessos definindo `backend spaceReserve parameter` como volume.

Como posso garantir que os volumes em uso não sejam excluídos mesmo se eu excluir acidentalmente o PVC?

A proteção de PVC é ativada automaticamente no Kubernetes a partir da versão 1.10.

Posso expandir os PVCs NFS que foram criados pelo Trident?

Sim. Você pode expandir um PVC criado pelo Trident. Observe que volume com crescimento automático é um recurso do ONTAP que não se aplica ao Trident.

Posso importar um volume enquanto ele estiver em SnapMirror Data Protection (DP) ou modo offline?

A importação do volume falha se o volume externo estiver no modo DP ou estiver offline. Você recebe a seguinte mensagem de erro:

```
Error: could not import volume: volume import failed to get size of
volume: volume <name> was not found (400 Bad Request) command terminated
with exit code 1.
Make sure to remove the DP mode or put the volume online before importing
the volume.
```

Como a quota de recursos é traduzida para um cluster NetApp?

A cota de recursos de armazenamento do Kubernetes deve funcionar enquanto o armazenamento NetApp tiver capacidade. Quando o armazenamento NetApp não consegue respeitar as configurações de cota do

Kubernetes devido à falta de capacidade, Trident tenta provisionar, mas ocorre um erro.

Posso criar Volume Snapshots usando Trident?

Sim. A criação de snapshots de volume sob demanda e Volumes Persistentes a partir de Snapshots são suportados pelo Trident. Para criar PVs a partir de snapshots, certifique-se de que o `VolumeSnapshotDataSource` feature gate foi habilitado.

Quais são os drivers que suportam snapshots de volume do Trident?

A partir de hoje, o suporte a snapshots sob demanda está disponível para nossos `ontap-nas`, `ontap-nas-flexgroup`, `ontap-san`, `ontap-san-economy`, `solidfire-san` e `azure-netapp-files` drivers de backend.

Como faço para criar um backup snapshot de um volume provisionado pelo Trident com ONTAP?

Isso está disponível nos `ontap-nas`, `ontap-san` e `ontap-nas-flexgroup` drivers. Você também pode especificar um `snapshotPolicy` para o `ontap-san-economy` driver no nível de FlexVol.

Isso também está disponível nos `ontap-nas-economy` drivers, mas na granularidade do nível de FlexVol volume e não na granularidade do nível de qtree. Para habilitar a capacidade de criar snapshots de volumes provisionados pelo Trident, defina a opção de parâmetro de backend `snapshotPolicy` para a política de snapshot desejada, conforme definida no backend do ONTAP. Quaisquer snapshots criados pelo controlador de armazenamento não são conhecidos pelo Trident.

Posso definir uma porcentagem de reserva de snapshots para um volume provisionado através do Trident?

Sim, você pode reservar uma porcentagem específica de espaço de disco para armazenar as cópias de snapshots por meio do Trident, definindo o `snapshotReserve` atributo no arquivo de definição do backend. Se você configurou `snapshotPolicy` e `snapshotReserve` no arquivo de definição do backend, a porcentagem de reserva de snapshots será definida de acordo com a porcentagem de `snapshotReserve` mencionada no arquivo do backend. Se o número percentual de `snapshotReserve` não for mencionado, o ONTAP, por padrão, considera a porcentagem de reserva de snapshots como 5. Se a opção `snapshotPolicy` estiver definida como none, a porcentagem de reserva de snapshots será definida como 0.

Posso acessar diretamente o diretório de snapshot do volume e copiar arquivos?

Sim, você pode acessar o diretório de snapshots no volume provisionado pelo Trident definindo o parâmetro `snapshotDir` no arquivo de definição do backend.

Posso configurar SnapMirror para volumes através do Trident?

Atualmente, SnapMirror precisa ser configurado externamente usando a CLI do ONTAP ou o OnCommand System Manager.

Como faço para restaurar volumes persistentes para um snapshot específico do ONTAP?

Para restaurar um volume para um snapshot do ONTAP, execute as seguintes etapas:

1. Suspenda o pod da aplicação que está utilizando o Persistent volume.
2. Reverta para o snapshot necessário através do ONTAP CLI ou OnCommand System Manager.
3. Reinicie o pod da aplicação.

O Trident pode provisionar volumes em SVMs que tenham um Load-Sharing Mirror configurado?

É possível criar espelhos de balanceamento de carga para volumes raiz de SVMs que fornecem dados via NFS. ONTAP atualiza automaticamente os espelhos de balanceamento de carga para volumes que foram criados pelo Trident. Isso pode resultar em atrasos na montagem dos volumes. Quando vários volumes são criados usando Trident, o provisionamento de um volume depende da atualização do espelho de balanceamento de carga pelo ONTAP.

Como posso separar o uso de storage class para cada cliente/inquilino?

O Kubernetes não permite classes de armazenamento em namespaces. No entanto, você pode usar o Kubernetes para limitar o uso de uma classe de armazenamento específica por namespace usando cotas de recursos de armazenamento, que são por namespace. Para negar a um namespace específico o acesso a um armazenamento específico, defina a cota de recursos como 0 para essa classe de armazenamento.

Solução de problemas

Utilize as dicas fornecidas aqui para solucionar problemas que você possa encontrar durante a instalação e o uso de Trident.

> Para obter ajuda com Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie-o para o Suporte da NetApp.

Solução de problemas gerais

- Se o Trident pod não iniciar corretamente (por exemplo, quando o Trident pod estiver preso na `ContainerCreating` fase com menos de dois contêineres prontos), executar `kubectl -n trident describe deployment trident` e `kubectl -n trident describe pod trident-**` pode fornecer informações adicionais. Obter os logs do kubelet (por exemplo, via `journalctl -xeu kubelet`) também pode ser útil.
- Se não houver informações suficientes nos logs do Trident, você pode tentar ativar o modo de depuração do Trident passando o `-d` parâmetro para o comando de instalação, de acordo com a opção de instalação escolhida.

  Em seguida, confirme se o modo de depuração está ativado usando `./tridentctl logs -n trident` e procurando por `level=debug msg` no log.

  Instalado com Operator

  ```
  kubectl patch torc trident -n <namespace> --type=merge -p
  '{"spec":{"debug":true}}'
  ```

  Isso vai reiniciar todos os pods do Trident, o que pode levar vários segundos. Você pode verificar isso observando a coluna 'AGE' na saída de `kubectl get pod -n trident`.

  Para Trident 20.07 e 20.10 use `tprov` em vez de `torc`.

  Instalado com Helm

```
helm upgrade <name> trident-operator-21.07.1-custom.tgz --set
tridentDebug=true`
```

Instalado com tridentctl

```
./tridentctl uninstall -n trident
./tridentctl install -d -n trident
```

- Você também pode obter logs de depuração para cada backend incluindo `debugTraceFlags` na definição do backend. Por exemplo, inclua `debugTraceFlags: {"api":true, "method":true,}` para obter chamadas de API e travessias de métodos nos logs do Trident. Backends existentes podem ter `debugTraceFlags` configurado com um `tridentctl backend update`.
- Ao usar o Red Hat Enterprise Linux CoreOS (RHCOS), certifique-se de que `iscsid` esteja habilitado nos nós de trabalho e iniciado por padrão. Isso pode ser feito usando OpenShift MachineConfigs ou modificando os modelos de ignição.
- Um problema comum que você pode encontrar ao usar Trident "Azure NetApp Files" é quando os segredos do locatário e do cliente vêm de um registro de aplicativo com permissões insuficientes. Para obter uma lista completa dos requisitos do Trident, consulte a "Azure NetApp Files"configuração.
- Se houver problemas ao montar um PV em um contêiner, verifique se `rpcbind` está instalado e em execução. Use o gerenciador de pacotes necessário para o sistema operacional do host e verifique se `rpcbind` está em execução. Você pode verificar o status do serviço `rpcbind` executando um `systemctl status rpcbind` ou equivalente.
- Se um backend do Trident reportar que está no estado `failed`, mesmo tendo funcionado anteriormente, é provável que a causa seja a alteração das credenciais de SVM/admin associadas ao backend. Atualizar as informações do backend usando `tridentctl update backend` ou reiniciar o pod do Trident corrigirá esse problema.
- Se você encontrar problemas de permissão ao instalar Trident com Docker como o runtime de contêiner, tente a instalação do Trident com a `--in cluster=false` flag. Isso não usará um pod de instalação e evitará problemas de permissão causados pelo `trident-installer` usuário.
- Use o `uninstall parameter <Uninstalling Trident>` para limpar após uma execução com falha. Por padrão, o script não remove os CRDs que foram criados pelo Trident, tornando seguro desinstalar e instalar novamente mesmo em uma implantação em execução.
- Se você deseja fazer downgrade para uma versão anterior do Trident, primeiro execute o comando `tridentctl uninstall` para remover Trident. Baixe o "Versão do Trident" desejado e instale usando o comando `tridentctl install`.
- Após uma instalação bem-sucedida, se um PVC ficar preso na `Pending` fase, executar `kubectl describe pvc` pode fornecer informações adicionais sobre por que Trident não conseguiu provisionar um PV para esse PVC.

Implantação malsucedida do Trident usando o operador

Se você estiver implantando Trident usando o operador, o status de `TridentOrchestrator` muda de `Installing` para `Installed`. Se você observar o status `Failed` e o operador não conseguir se recuperar sozinho, você deve verificar os logs do operador executando o seguinte comando:

```
tridentctl logs -l trident-operator
```

Analisar os logs do contêiner trident-operator pode indicar a origem do problema. Por exemplo, um problema possível poderia ser a impossibilidade de obter as imagens de contêiner necessárias dos registros upstream em um ambiente airgapped.

Para entender por que a instalação do Trident não foi bem-sucedida, você deve verificar o `TridentOrchestrator` status.

```
kubectl describe torc trident-2
Name:         trident-2
Namespace:
Labels:       <none>
Annotations:  <none>
API Version:  trident.netapp.io/v1
Kind:         TridentOrchestrator
...
Status:
  Current Installation Params:
    IPv6:
    Autosupport Hostname:
    Autosupport Image:
    Autosupport Proxy:
    Autosupport Serial Number:
    Debug:
    Image Pull Secrets:         <nil>
    Image Registry:
    k8sTimeout:
    Kubelet Dir:
    Log Format:
    Silence Autosupport:
    Trident Image:
  Message:                      Trident is bound to another CR 'trident'
  Namespace:                    trident-2
  Status:                       Error
  Version:
Events:
  Type     Reason  Age                From                          Message
  ----     ------  ----               ----                          -------
  Warning  Error   16s (x2 over 16s)  trident-operator.netapp.io  Trident
is bound to another CR 'trident'
```

Este erro indica que já existe uma `TridentOrchestrator` que foi usada para instalar o Trident. Como cada cluster Kubernetes só pode ter uma instância do Trident, o operador garante que, a qualquer momento, exista apenas uma `TridentOrchestrator` ativa que ele pode criar.

Além disso, observar o status dos pods Trident pode muitas vezes indicar se algo não está certo.

```
kubectl get pods -n trident

NAME                                READY   STATUS             RESTARTS
AGE
trident-csi-4p5kq                   1/2     ImagePullBackOff   0
5m18s
trident-csi-6f45bfd8b6-vfrkw        4/5     ImagePullBackOff   0
5m19s
trident-csi-9q5xc                   1/2     ImagePullBackOff   0
5m18s
trident-csi-9v95z                   1/2     ImagePullBackOff   0
5m18s
trident-operator-766f7b8658-ldzsv   1/1     Running            0
8m17s
```

É possível ver claramente que os pods não conseguem inicializar completamente porque uma ou mais imagens de contêiner não foram buscadas.

Para resolver o problema, você deve editar o `TridentOrchestrator` CR. Alternativamente, você pode excluir `TridentOrchestrator` e criar um novo com a definição modificada e precisa.

Implantação malsucedida do Trident usando `tridentctl`

Para ajudar a descobrir o que deu errado, você pode executar o instalador novamente usando o `-d` argumento, o que ativará o modo de depuração e ajudará você a entender qual é o problema:

```
./tridentctl install -n trident -d
```

Após resolver o problema, você pode limpar a instalação da seguinte forma e, em seguida, executar o comando `tridentctl install` novamente:

```
./tridentctl uninstall -n trident
INFO Deleted Trident deployment.
INFO Deleted cluster role binding.
INFO Deleted cluster role.
INFO Deleted service account.
INFO Removed Trident user from security context constraint.
INFO Trident uninstallation succeeded.
```

Remova completamente Trident e os CRDs

Você pode remover completamente Trident e todos os CRDs criados e os recursos personalizados associados.

Esta ação não pode ser desfeita. Não faça isso a menos que deseje uma instalação completamente nova do Trident. Para desinstalar Trident sem remover os CRDs, consulte "Desinstalar Trident".

Operador Trident

Para desinstalar Trident e remover completamente os CRDs usando o operador Trident:

```
kubectl patch torc <trident-orchestrator-name> --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

Helm

Para desinstalar Trident e remover completamente os CRDs usando Helm:

```
kubectl patch torc trident --type=merge -p
'{"spec":{"wipeout":["crds"],"uninstall":true}}'
```

<code>tridentctl</code>

Para remover completamente os CRDs após desinstalar Trident usando `tridentctl`

```
tridentctl obliviate crd
```

Falha no desempacotamento de nó NVMe com namespaces de bloco bruto RWX no Kubernetes 1.26

Se você estiver executando o Kubernetes 1.26, a desestabilização de nós pode falhar ao usar NVMe/TCP com namespaces de bloco bruto RWX. Os seguintes cenários fornecem uma solução alternativa para a falha. Como alternativa, você pode atualizar o Kubernetes para 1.27.

Excluídos o namespace e o pod

Considere um cenário em que você tenha um namespace gerenciado pelo Trident (volume persistente NVMe) anexado a um pod. Se você excluir o namespace diretamente do backend ONTAP, o processo de descompactação ficará travado após você tentar excluir o pod. Esse cenário não afeta o cluster Kubernetes nem outras funcionalidades.

Solução alternativa

Desmonte o volume persistente (correspondente a esse namespace) do respectivo nó e exclua-o.

LIFs de dados bloqueados

```
 If you block (or bring down) all the dataLIFs of the NVMe Trident
backend, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Traga o dataLIFS de volta para restaurar a funcionalidade completa.
```

Mapeamento de namespace excluído

```
 If you remove the `hostNQN` of the worker node from the corresponding
subsystem, the unstaging process gets stuck when you attempt to delete the
pod. In this scenario, you cannot run any NVMe CLI commands on the
Kubernetes node.
.Solução alternativa
Adicione o  `hostNQN` de volta ao subsistema.
```

Clientes NFSv4.2 relatam "argumento inválido" após atualizar o ONTAP quando esperam que o "v4.2-xattrs" esteja habilitado

Após a atualização do ONTAP, os clientes NFSv4.2 podem apresentar erros de "argumento inválido" ao tentar montar exportações NFSv4.2. Esse problema ocorre quando a `v4.2-xattrs` opção não está habilitada na SVM. Solução alternativa: habilite a `v4.2-xattrs` opção na SVM ou atualize para ONTAP 9.12.1 ou posterior, onde essa opção já está habilitada por padrão.

Suporte

NetApp oferece suporte para Trident de diversas maneiras. Amplas opções de autoatendimento gratuitas estão disponíveis 24x7, como artigos da knowledgebase (KB) e um canal no Discord.

Ciclo de vida do Trident

Trident oferece três níveis de suporte com base na sua versão. Consulte "NetApp suporte de versão de software para definições".

Suporte total
: Trident oferece suporte completo por doze meses a partir da data de lançamento.

Suporte limitado
: Trident oferece suporte limitado para os meses 13 a 24 a partir da data de lançamento.

Autoatendimento
: A documentação do Trident estará disponível para os meses 25 a 36 a partir da data de lançamento.

Versão	Suporte total	Suporte limitado	Autoatendimento

"25,10"	Outubro 2026	Outubro 2027	Outubro 2028
"25,06"	Junho de 2026	Junho de 2027	Junho de 2028
"25,02"	Fevereiro de 2026	Fevereiro de 2027	Fevereiro de 2028
"24,10"	—	Outubro 2026	Outubro 2027
"24,06"	—	Junho de 2026	Junho de 2027
"24,02"	—	Fevereiro de 2026	Fevereiro de 2027
"23,10"	—	—	Outubro 2026
"23,07"	—	—	Julho de 2026
"23,04"	—	—	Abril de 2026
"23,01"	—	—	Janeiro de 2026

Autoatendimento

Para obter uma lista completa de artigos de resolução de problemas, consulte "NetApp Knowledgebase (login necessário)".

Suporte da comunidade

Existe uma comunidade pública ativa de usuários de contêineres (incluindo desenvolvedores do Trident) em nosso "Canal do Discord". Este é um ótimo lugar para fazer perguntas gerais sobre o projeto e discutir tópicos relacionados com colegas que compartilham os mesmos interesses.

NetApp suporte técnico da NetApp

Para obter ajuda com Trident, crie um pacote de suporte usando `tridentctl logs -a -n trident` e envie-o para `NetApp Support <Getting Help>`.

Para obter mais informações

- "Recursos Trident"
- "Kubernetes Hub"

Referência

Portas Trident

Saiba mais sobre as portas que Trident utiliza para comunicação.

Visão geral

Trident utiliza diversas portas para comunicação dentro de clusters Kubernetes e com storage backends. A seguir, um resumo das principais portas, suas finalidades e considerações de segurança.

- **Foco em tráfego de saída**: Os nós do Kubernetes (controller e worker) iniciam principalmente o tráfego para LIFs/IPs de storage, portanto, as regras do iptables devem permitir tráfego de saída dos IPs dos nós para IPs de storage específicos nessas portas. Evite regras amplas do tipo "qualquer para qualquer".
- **Restrições de entrada**: limite as portas internas do Trident ao tráfego interno do cluster (por exemplo, usando CNI como Calico). Nenhuma exposição desnecessária de entrada nos firewalls do host.
- **Segurança de protocolo**:
  - Use TCP sempre que possível (mais confiável).
  - Habilite CHAP/IPsec para iSCSI se sensível; TLS/HTTPS para gerenciamento (porta 443/8443).
  - Para NFSv4 (padrão no Trident), remova as portas UDP/NFSv3 mais antigas (por exemplo, 4045-4049) se não forem necessárias.
  - Restrinja o acesso a sub-redes confiáveis; monitore com ferramentas como Prometheus (porta 8001 opcional).

Portas para nós controladores

Essas portas são destinadas principalmente ao Trident operator (gerenciamento de backend). Todas as portas internas são de nível de pod; permita nos nós somente se o firewall do host interferir com o CNI.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 8000	Entrada/Saída (cluster-internal)	Servidor REST Trident (comunicação operador-controlador)	Todos	Restringir aos CIDRs dos pods; sem exposição externa.
TCP 8443	Entrada/Saída (cluster-internal)	Backchannel HTTPS (API interna segura)	Todos	Criptografia TLS; limite ao service mesh do Kubernetes se utilizado.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Todos	Exponha apenas às ferramentas de monitoramento (por exemplo, usando RBAC); desative se não for usado.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Exigir validação de certificado TLS; restringir apenas aos IPs de mgmt do LIF.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	API REST padrão; utiliza certificados; configurável no YAML do backend.

Portas para nós de trabalho

Essas portas são para daemonsets de nós CSI e montagens de pods. As portas de dados são de saída para LIFs de dados de storage; inclua extras do NFSv3 se estiver usando NFSv3 (opcional para NFSv4).

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
TCP 17546	Entrada (local para pod)	Sondas de liveness/prontidão do nó CSI	Todos	Configurável (--probe-port); garanta que não haja conflitos de host; somente local.
TCP 8000	Entrada/Saída (cluster-internal)	Servidor REST Trident	Todos	Como acima; interno ao pod.
TCP 8443	Entrada/Saída (cluster-internal)	HTTPS de canal de retorno	Todos	Como acima.
TCP 8001	Entrada (interna ao cluster, opcional)	Métricas Prometheus	Todos	Como acima.
TCP 443	Saída	HTTPS para ONTAP SVM/cluster mgmt LIF	ONTAP (todos), ANF	Conforme mencionado acima; usado para descoberta.
TCP 8443	Saída	HTTPS para Proxy de Serviços Web E-Series	E-Series (iSCSI)	Como acima.
TCP/UDP 111	Saída	RPCBIND/portmapper	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Obrigatório para v3; opcional para v4 (descarregamento do firewall); restringir se estiver usando somente NFSv4.
TCP/UDP 2049	Saída	Daemon NFS	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Dados essenciais; bem conhecidos; use TCP para confiabilidade.
TCP/UDP 635	Saída	Daemon de montagem	ONTAP-NAS (NFSv3/v4), ANF (NFS)	Montagem; callbacks bidirecionais possíveis (permitir entrada efêmera se necessário).
UDP 4045	Saída	Gerenciador de bloqueio NFS (nlockmgr)	ONTAP-NAS (NFSv3)	Bloqueio de arquivos; ignorar para v4 (pNFS handles); somente UDP.

Porta/Protocolo	Direção	Propósito	Driver/Protocolo	Notas de segurança
UDP 4046	Saída	Monitor de status NFS (statd)	ONTAP-NAS (NFSv3)	Notificações; podem ser necessárias portas efêmeras de entrada (1024-65535) para callbacks.
UDP 4049	Saída	Daemon de cotas NFS (rquotad)	ONTAP-NAS (NFSv3)	Cotas; pular para v4.
TCP 3260	Saída	destino iSCSI (descoberta/dados/CHAP)	ONTAP-SAN (iSCSI), E-Series (iSCSI)	Bem conhecido; autenticação CHAP nesta porta; habilite CHAP mútuo para segurança.
TCP 445	Saída	SMB/CIFS	ONTAP-NAS (SMB), ANF (SMB)	Bem conhecido; use SMB3 com criptografia (Trident annotation netapp.io/smb-encryption=true).
TCP/UDP 88 (opcional)	Saída	Autenticação Kerberos	ONTAP (NFS/SMB/iSCSI com Kerberos)	Se estiver usando Kerberos (não é o padrão); para servidores AD, não para storage.
TCP/UDP 389 (opcional)	Saída	LDAP	ONTAP (NFS/SMB com LDAP)	Semelhante; para resolução de nomes/autenticação; restringir ao AD.

A porta de liveness/readiness pode ser alterada durante a instalação usando a flag `--probe-port`. É importante garantir que essa porta não esteja sendo usada por outro processo nos nós de trabalho.

API REST Trident

Embora "Comandos e opções do tridentctl" sejam a maneira mais fácil de interagir com a Trident API REST, você pode usar o endpoint REST diretamente se preferir.

Quando usar a API REST

API REST é útil para instalações avançadas que usam Trident como um binário independente em implantações não-Kubernetes.

Para maior segurança, o Trident `REST API` é restrito ao localhost por padrão quando executado dentro de um pod. Para alterar esse comportamento, você precisa definir o argumento do Trident `-address` em sua configuração de pod.

Usando API REST

Para exemplos de como essas APIs são chamadas, passe a flag (`-d` de debug. Para mais informações, consulte "Gerencie Trident usando tridentctl".

A API funciona da seguinte forma:

GET

`GET <trident-address>/trident/v1/<object-type>`

Lista todos os objetos desse tipo.

`GET <trident-address>/trident/v1/<object-type>/<object-name>`

Obtém os detalhes do objeto nomeado.

POST

`POST <trident-address>/trident/v1/<object-type>`

Cria um objeto do tipo especificado.

- Requer uma configuração JSON para o objeto a ser criado. Para a especificação de cada tipo de objeto, consulte "Gerencie Trident usando tridentctl".
- Se o objeto já existir, o comportamento varia: backends atualizam o objeto existente, enquanto todos os outros tipos de objeto falharão a operação.

EXCLUIR

`DELETE <trident-address>/trident/v1/<object-type>/<object-name>`

Exclui o recurso nomeado.

> Os volumes associados a backends ou classes de armazenamento continuarão a existir; estes devem ser excluídos separadamente. Para obter mais informações, consulte "Gerencie Trident usando tridentctl".

Opções de linha de comando

Trident expõe diversas opções de linha de comando para o orquestrador Trident. Você pode usar essas opções para modificar sua implantação.

Log

`-debug`

Ativa a saída de depuração.

`-loglevel <level>`

Define o nível de registro (debug, info, warn, error, fatal). O padrão é info.

Kubernetes

`-k8s_pod`

Use esta opção ou `-k8s_api_server` para habilitar o suporte ao Kubernetes. Ao configurar esta opção, Trident utiliza as credenciais da conta de serviço do Kubernetes do pod que o contém para contatar o servidor da API. Isso só funciona quando Trident é executado como um pod em um cluster Kubernetes com contas de serviço habilitadas.

-k8s_api_server <insecure-address:insecure-port>

Use esta opção ou `-k8s_pod` para habilitar o suporte ao Kubernetes. Quando especificado, Trident se conecta ao servidor da API do Kubernetes usando o endereço e a porta inseguros fornecidos. Isso permite que o Trident seja implantado fora de um pod; no entanto, ele suporta apenas conexões inseguras com o servidor da API. Para se conectar com segurança, implante Trident em um pod com a opção `-k8s_pod`.

Docker

-volume_driver <name>

Nome do driver usado ao registrar o plugin do Docker. Padrão: `netapp`.

-driver_port <port-number>

Escute nesta porta em vez de um socket de domínio UNIX.

-config <file>

Obrigatório; você deve especificar este caminho para um arquivo de configuração de backend.

REST

-address <ip-or-host>

Especifica o endereço no qual o servidor REST do Trident deve escutar. O padrão é localhost. Quando escuta em localhost e está em execução dentro de um pod do Kubernetes, a interface REST não é diretamente acessível de fora do pod. Use `-address ""` para tornar a interface REST acessível a partir do endereço IP do pod.

> A interface REST do Trident pode ser configurada para escutar e servir apenas em 127.0.0.1 (para IPv4) ou [::1] (para IPv6).

-port <port-number>

Especifica a porta na qual o servidor REST do Trident deve escutar. O padrão é 8000.

-rest

Habilita a interface REST. O valor padrão é true.

Objetos Kubernetes e Trident

Você pode interagir com Kubernetes e Trident usando APIs REST, lendo e gravando objetos de recursos. Existem diversos objetos de recursos que definem a relação entre Kubernetes e Trident, Trident e storage, e Kubernetes e storage. Alguns desses objetos são gerenciados por meio do Kubernetes e outros são gerenciados por meio do Trident.

Como os objetos interagem entre si?

Talvez a maneira mais fácil de entender os objetos, para que servem e como interagem, seja acompanhar uma única solicitação de storage de um usuário do Kubernetes:

1. Um usuário cria uma `PersistentVolumeClaim` solicitando um novo `PersistentVolume` de um tamanho específico de um `StorageClass` Kubernetes que foi previamente configurado pelo administrador.

2. O Kubernetes `StorageClass` identifica Trident como seu provisionador e inclui parâmetros que informam Trident como provisionar um volume para a classe solicitada.
3. Trident analisa seus próprios `StorageClass` com o mesmo nome que identifica a correspondência `Backends` e `StoragePools` que pode usar para provisionar volumes para a classe.
4. Trident provisiona storage em um backend correspondente e cria dois objetos: um `PersistentVolume` no Kubernetes que informa ao Kubernetes como encontrar, montar e tratar o volume, e um volume no Trident que mantém a relação entre o `PersistentVolume` e o storage real.
5. O Kubernetes vincula o `PersistentVolumeClaim` ao novo `PersistentVolume`. Os pods que incluem o `PersistentVolumeClaim` montam esse PersistentVolume em qualquer host em que sejam executados.
6. Um usuário cria um `VolumeSnapshot` de um PVC existente, usando um `VolumeSnapshotClass` que aponta para Trident.
7. Trident identifica o volume associado ao PVC e cria um snapshot do volume em seu backend. Ele também cria um `VolumeSnapshotContent` que instrui o Kubernetes sobre como identificar o snapshot.
8. Um usuário pode criar um `PersistentVolumeClaim` usando `VolumeSnapshot` como a fonte.
9. Trident identifica o snapshot necessário e executa o mesmo conjunto de etapas envolvidas na criação de `PersistentVolume` e de `Volume`.

Para obter mais informações sobre objetos do Kubernetes, recomendamos fortemente que você leia a "Volumes persistentes" seção da documentação do Kubernetes.

Objetos do Kubernetes `PersistentVolumeClaim`

Um objeto `PersistentVolumeClaim`Kubernetes é uma solicitação de armazenamento feita por um usuário do cluster Kubernetes.

Além da especificação padrão, Trident permite que os usuários especifiquem as seguintes anotações específicas de volume se quiserem substituir os valores padrão que você definiu na configuração do backend:

Anotação	Opção de volume	Drivers com suporte
trident.netapp.io/fileSystem	fileSystem	ontap-san, solidfire-san,ontap-san-economy
trident.netapp.io/cloneFromPVC	cloneSourceVolume	ontap-nas, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy
trident.netapp.io/splitOnClone	splitOnClone	ontap-nas, ontap-san
trident.netapp.io/protocol	protocolo	qualquer
trident.netapp.io/exportPolicy	exportPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/snapshotPolicy	snapshotPolicy	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san
trident.netapp.io/snapshotReserve	snapshotReserve	ontap-nas, ontap-nas-flexgroup, ontap-san

Anotação	Opção de volume	Drivers com suporte
trident.netapp.io/snapshotDirectory	snapshotDirectory	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/unixPermissions	unixPermissions	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup
trident.netapp.io/blockSize	blockSize	solidfire-san
trident.netapp.io/skipRecoveryQueue	skipRecoveryQueue	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy

Se o PV criado tiver a `Delete` política de recuperação, Trident exclui tanto o PV quanto o volume de suporte quando o PV é liberado (ou seja, quando o usuário exclui o PVC). Caso a ação de exclusão falhe, Trident marca o PV como tal e tenta a operação periodicamente até que seja bem-sucedida ou o PV seja excluído manualmente. Se o PV usar a `Retain` política, Trident a ignora e assume que o administrador o removerá do Kubernetes e do backend, permitindo que o volume seja copiado ou inspecionado antes de sua remoção. Observe que a exclusão do PV não faz com que Trident exclua o volume de suporte. Você deve removê-lo usando a API REST (`tridentctl`.

Trident suporta a criação de Snapshots de Volume usando a especificação CSI: você pode criar um Snapshot de Volume e usá-lo como uma fonte de dados para clonar PVCs existentes. Dessa forma, cópias ponto no tempo de PVs podem ser expostas ao Kubernetes na forma de snapshots. Os snapshots podem então ser usados para criar novos PVs. Veja `On-Demand Volume Snapshots` para ver como isso funcionaria.

Trident também fornece as `cloneFromPVC` e `splitOnClone`anotações para criar clones. Você pode usar essas anotações para clonar um PVC sem precisar usar a implementação CSI.

Aqui está um exemplo: se um usuário já possui um PVC chamado `mysql`, o usuário pode criar um novo PVC chamado `mysqlclone` usando a anotação, como `trident.netapp.io/cloneFromPVC: mysql`. Com essa anotação definida, Trident clona o volume correspondente ao PVC mysql, em vez de provisionar um volume do zero.

Considere os seguintes pontos:

- NetApp recomenda clonar um volume ocioso.
- Um PVC e seu clone devem estar no mesmo namespace do Kubernetes e ter a mesma storage class.
- Com os `ontap-nas` e `ontap-san` drivers, pode ser desejável definir a anotação PVC `trident.netapp.io/splitOnClone` em conjunto com `trident.netapp.io/cloneFromPVC`. Com `trident.netapp.io/splitOnClone` definido como `true`, o Trident divide o volume clonado do volume pai, desacoplando completamente o ciclo de vida do volume clonado do seu volume pai, à custa de alguma perda de eficiência de storage. Não definir `trident.netapp.io/splitOnClone` ou defini-lo como `false` resulta em menor consumo de espaço no backend, à custa de criar dependências entre os volumes pai e clone, de forma que o volume pai não possa ser excluído a menos que o clone seja excluído primeiro. Um cenário em que dividir o clone faz sentido é clonar um volume de banco de dados vazio, onde se espera que o volume e seu clone divirjam bastante e não se beneficiem das eficiências de storage oferecidas pelo ONTAP.

O `sample-input` diretório contém exemplos de definições de PVC para uso com Trident. Consulte para obter uma descrição completa dos parâmetros e configurações associados aos volumes do Trident.

Objetos do Kubernetes `PersistentVolume`

Um objeto Kubernetes `PersistentVolume` representa um espaço de armazenamento disponibilizado para o cluster Kubernetes. Ele possui um ciclo de vida independente do pod que o utiliza.

> Trident cria `PersistentVolume` objetos e os registra automaticamente no cluster Kubernetes com base nos volumes que ele provisiona. Não é necessário que você os gerencie.

Ao criar um PVC que se refere a um Trident-based `StorageClass`, Trident provisiona um novo volume usando a classe de armazenamento correspondente e registra um novo PV para esse volume. Ao configurar o volume provisionado e o PV correspondente, Trident segue as seguintes regras:

- Trident gera um nome de PV para Kubernetes e um nome interno que utiliza para provisionar o storage. Em ambos os casos, garante que os nomes sejam únicos em seu escopo.
- O tamanho do volume corresponde o mais próximo possível ao tamanho solicitado no PVC, embora possa ser arredondado para a quantidade alocável mais próxima, dependendo da plataforma.

Objetos do Kubernetes `StorageClass`

Objetos do Kubernetes `StorageClass` são especificados por nome em `PersistentVolumeClaims` para provisionar armazenamento com um conjunto de propriedades. A própria storage class identifica o provisionador a ser usado e define esse conjunto de propriedades em termos que o provisionador entende.

É um dos dois objetos básicos que precisam ser criados e gerenciados pelo administrador. O outro é o objeto de backend do Trident.

Um objeto Kubernetes `StorageClass` que usa Trident tem a seguinte aparência:

```
apiVersion: storage.k8s.io/v1
kind: StorageClass
metadata:
  name: <Name>
provisioner: csi.trident.netapp.io
mountOptions: <Mount Options>
parameters: <Trident Parameters>
allowVolumeExpansion: true
volumeBindingMode: Immediate
```

Esses parâmetros são específicos do Trident e informam ao Trident como provisionar volumes para a classe.

Os parâmetros da classe de armazenamento são:

Atributo	Tipo	Obrigatório	Descrição
atributos	map[string]string	não	Consulte a seção de atributos abaixo
storagePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro

Atributo	Tipo	Obrigatório	Descrição
additionalStoragePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro
excludeStoragePools	map[string]StringList	não	Mapeamento de nomes de backend para listas de pools de storage dentro

Os atributos de armazenamento e seus possíveis valores podem ser classificados em atributos de seleção de pool de storage e atributos do Kubernetes.

Atributos de seleção do storage pool

Esses parâmetros determinam quais pools de storage gerenciados pelo Trident devem ser utilizados para provisionar volumes de um determinado tipo.

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
mídia[1]	string	hdd, híbrido, ssd	O pool contém mídias deste tipo; híbrido significa ambos	Tipo de mídia especificado	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san
provisioningType	string	fino, grosso	Pool suporta este método de provisionamento	Método de provisionamento especificado	espesso: all ontap; fino: all ontap & solidfire-san
backendType	string	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, solidfire-san, azure-netapp-files, ontap-san-economy	Pool pertence a este tipo de backend	Backend especificado	Todos os drivers
instantâneos	bool	true, false	O pool suporta volumes com snapshots	Volume com snapshots ativados	ontap-nas, ontap-san, solidfire-san
clones	bool	true, false	Pool suporta clonagem de volumes	Volume com clonagem ativada	ontap-nas, ontap-san, solidfire-san
criptografia	bool	true, false	Pool suporta volumes criptografados	Volume com criptografia ativada	ontap-nas, ontap-nas-economy, ontap-nas-flexgroups, ontap-san

Atributo	Tipo	Valores	Oferta	Solicitação	Apoiado por
IOPS	inteiro	inteiro positivo	O pool é capaz de garantir IOPS nessa faixa	Volume garantiu esses IOPS	solidfire-san

[1]: não suportado pelos sistemas ONTAP Select

Na maioria dos casos, os valores solicitados influenciam diretamente o provisionamento; por exemplo, solicitar provisionamento espesso resulta em um volume provisionado de forma espessa. No entanto, um pool de storage usa seus valores mínimo e máximo de IOPS oferecidos para definir os valores de QoS, em vez do valor solicitado. Nesse caso, o valor solicitado é usado apenas para selecionar o pool de storage.

Idealmente, você pode usar `attributes` sozinho para modelar as qualidades do armazenamento necessário para atender às necessidades de uma classe específica. Trident descobre e seleciona automaticamente pools de storage que correspondem a *todos* os `attributes` que você especificar.

Caso você não consiga usar `attributes` para selecionar automaticamente os pools corretos para uma classe, você pode usar os parâmetros `storagePools` e `additionalStoragePools` para refinar ainda mais os pools ou até mesmo para selecionar um conjunto específico de pools.

Você pode usar o `storagePools` parâmetro para restringir ainda mais o conjunto de pools que correspondem a qualquer `attributes` especificado. Em outras palavras, Trident usa a interseção dos pools identificados pelos parâmetros `attributes` e `storagePools` para provisionamento. Você pode usar qualquer um dos parâmetros sozinho ou ambos juntos.

Você pode usar o `additionalStoragePools` parâmetro para estender o conjunto de pools que Trident usa para provisionamento, independentemente de quaisquer pools selecionados pelos `attributes` e `storagePools` parâmetros.

Você pode usar o `excludeStoragePools`parâmetro para filtrar o conjunto de pools que Trident usa para provisionamento. Usar esse parâmetro remove quaisquer pools correspondentes.

Nos `storagePools` e `additionalStoragePools` parâmetros, cada entrada assume o formato `<backend>:<storagePoolList>`, onde `<storagePoolList>` é uma lista de pools de storage separados por vírgula para o backend especificado. Por exemplo, um valor para `additionalStoragePools` pode ser algo como `ontapnas_192.168.1.100:aggr1,aggr2;solidfire_192.168.1.101:bronze`. Essas listas aceitam valores regex tanto para o backend quanto para os valores da lista. Você pode usar `tridentctl get backend` para obter a lista de backends e seus pools.

Atributos do Kubernetes

Esses atributos não têm impacto na seleção de pools/backends de storage pelo Trident durante o provisionamento dinâmico. Em vez disso, esses atributos simplesmente fornecem parâmetros suportados pelos Volumes Persistentes do Kubernetes. Os nós de trabalho são responsáveis pelas operações de criação do sistema de arquivos e podem exigir utilitários de sistema de arquivos, como xfsprogs.

Atributo	Tipo	Valores	Descrição	Drivers relevantes	Versão do Kubernetes
fsType	string	ext4, ext3, xfs	O tipo de sistema de arquivos para volumes em bloco	solidfire-san, ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy	Todos
allowVolumeExpansion	booleano	true, false	Ativar ou desativar o suporte para aumentar o tamanho do PVC	ontap-nas, ontap-nas-economy, ontap-nas-flexgroup, ontap-san, ontap-san-economy, solidfire-san, azure-netapp-files	1.11+
volumeBindingMode	string	Imediato, WaitForFirstConsumer	Escolha quando a vinculação de volume e o provisionamento dinâmico ocorre	Todos	1.19 - 1.26

- O `fsType` parâmetro é usado para controlar o tipo de sistema de arquivos desejado para LUNs SAN. Além disso, o Kubernetes também usa a presença de `fsType` em uma storage class para indicar que existe um sistema de arquivos. A propriedade do volume pode ser controlada usando o `fsGroup` security context de um pod somente se `fsType` estiver definido. Consulte "Kubernetes: configurar um contexto de segurança para um Pod ou contêiner" para uma visão geral sobre como definir a propriedade do volume usando o `fsGroup` context. O Kubernetes aplicará o valor `fsGroup` somente se:
  - `fsType` está definido na classe de armazenamento.
  - O modo de acesso do PVC é RWO.

  Para drivers de armazenamento NFS, um sistema de arquivos já existe como parte da exportação NFS. Para usar `fsGroup` a classe de armazenamento ainda é necessário especificar um `fsType`. Você pode defini-lo como `nfs` ou qualquer valor não nulo.
- Consulte "Expandir volumes" para obter mais detalhes sobre expansão de volume.
- O pacote de instalação do Trident fornece vários exemplos de definições de classes de armazenamento para uso com Trident `sample-input/storage-class-*.yaml`. A exclusão de uma classe de armazenamento do Kubernetes faz com que a classe de armazenamento correspondente do Trident também seja excluída.

Objetos do Kubernetes `VolumeSnapshotClass`

Objetos do Kubernetes `VolumeSnapshotClass` são análogos a `StorageClasses`. Eles ajudam a definir

várias classes de storage e são referenciados por snapshots de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de volume está associado a uma única classe de snapshot de volume.

A `VolumeSnapshotClass` deve ser definida por um administrador para criar snapshots. Uma classe de snapshot de volume é criada com a seguinte definição:

```
apiVersion: snapshot.storage.k8s.io/v1
kind: VolumeSnapshotClass
metadata:
  name: csi-snapclass
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações de snapshots de volume da `csi-snapclass` classe são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um snapshot precisa ser excluído. Quando `deletionPolicy` é definido como `Delete`, os objetos de snapshot de volume, bem como o snapshot subjacente no cluster de storage, são removidos quando um snapshot é excluído. Alternativamente, defini-lo como `Retain` significa que `VolumeSnapshotContent` e o snapshot físico são mantidos.

Objetos do Kubernetes `VolumeSnapshot`

Um objeto Kubernetes `VolumeSnapshot` é uma solicitação para criar um snapshot de um volume. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um snapshot de volume é uma solicitação feita por um usuário para criar um snapshot de um PVC existente.

Quando uma solicitação de snapshot de volume é recebida, Trident gerencia automaticamente a criação do snapshot para o volume no backend e expõe o snapshot criando um objeto `VolumeSnapshotContent` exclusivo. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> O ciclo de vida de um VolumeSnapshot é independente do PVC de origem: um snapshot persiste mesmo após a exclusão do PVC de origem. Ao excluir um PVC que possui snapshots associados, Trident marca o volume de suporte desse PVC no estado **Deleting**, mas não o remove completamente. O volume é removido quando todos os snapshots associados são excluídos.

Objetos do Kubernetes `VolumeSnapshotContent`

Um objeto `VolumeSnapshotContent`\`Kubernetes representa um snapshot obtido de um volume já provisionado. É análogo a um \`PersistentVolume` e indica um snapshot provisionado no cluster de armazenamento. Semelhante a `PersistentVolumeClaim` e `PersistentVolume` objetos, quando um snapshot é criado, o objeto `VolumeSnapshotContent` mantém um mapeamento um-para-um com o objeto `VolumeSnapshot` que solicitou a criação do snapshot.

O `VolumeSnapshotContent` objeto contém detalhes que identificam exclusivamente o snapshot, como o `snapshotHandle`. Esta `snapshotHandle` é uma combinação única do nome do PV e do nome do `VolumeSnapshotContent` objeto.

Quando uma solicitação de snapshot é recebida, Trident cria o snapshot no backend. Após a criação do

snapshot, Trident configura um `VolumeSnapshotContent` objeto e assim expõe o snapshot à API do Kubernetes.

> Normalmente, você não precisa gerenciar o `VolumeSnapshotContent` objeto. Uma exceção a isso é quando você deseja "importar um snapshot de volume" criado fora do Trident.

Kubernetes `VolumeGroupSnapshotClass` objetos

Objetos do Kubernetes `VolumeGroupSnapshotClass` são análogos a `VolumeSnapshotClass`. Eles ajudam a definir várias classes de storage e são referenciados por snapshots de grupo de volume para associar o snapshot à classe de snapshot necessária. Cada snapshot de grupo de volume está associado a uma única classe de snapshot de grupo de volume.

A `VolumeGroupSnapshotClass` deve ser definida por um administrador para criar um grupo de snapshots. Uma classe de snapshot de grupo de volumes é criada com a seguinte definição:

```
apiVersion: groupsnapshot.storage.k8s.io/v1beta1
kind: VolumeGroupSnapshotClass
metadata:
  name: csi-group-snap-class
  annotations:
    kubernetes.io/description: "Trident group snapshot class"
driver: csi.trident.netapp.io
deletionPolicy: Delete
```

O `driver` especifica ao Kubernetes que as solicitações para snapshots de grupo de volumes da classe `csi-group-snap-class` são tratadas pelo Trident. O `deletionPolicy` especifica a ação a ser tomada quando um snapshot de grupo precisa ser excluído. Quando `deletionPolicy` é definido como `Delete`, os objetos de snapshot do grupo de volumes, bem como o snapshot subjacente no cluster de storage, são removidos quando um snapshot é excluído. Alternativamente, defini-lo como `Retain` significa que `VolumeGroupSnapshotContent` e o snapshot físico são mantidos.

Objetos do Kubernetes `VolumeGroupSnapshot`

Um objeto do Kubernetes `VolumeGroupSnapshot` é uma solicitação para criar um snapshot de múltiplos volumes. Assim como um PVC representa uma solicitação feita por um usuário para um volume, um snapshot de grupo de volumes é uma solicitação feita por um usuário para criar um snapshot de um PVC existente.

Quando uma solicitação de snapshot de grupo de volumes é recebida, Trident gerencia automaticamente a criação do snapshot do grupo para os volumes no backend e expõe o snapshot criando um objeto `VolumeGroupSnapshotContent` exclusivo. Você pode criar snapshots a partir de PVCs existentes e usar os snapshots como DataSource ao criar novos PVCs.

> O ciclo de vida de um VolumeGroupSnapshot é independente do PVC de origem: um snapshot persiste mesmo após a exclusão do PVC de origem. Ao excluir um PVC que possui snapshots associados, Trident marca o volume de suporte desse PVC no estado **Deleting**, mas não o remove completamente. O snapshot do grupo de volumes é removido quando todos os snapshots associados são excluídos.

Kubernetes `VolumeGroupSnapshotContent` objetos

Um objeto Kubernetes `VolumeGroupSnapshotContent` representa um snapshot de grupo obtido de um volume já provisionado. É análogo a um `PersistentVolume` e indica um snapshot provisionado no cluster de storage. Semelhante a `PersistentVolumeClaim` e `PersistentVolume` objetos, quando um snapshot é criado, o objeto `VolumeSnapshotContent` mantém um mapeamento um-para-um com o objeto `VolumeSnapshot` que solicitou a criação do snapshot.

O `VolumeGroupSnapshotContent` objeto contém detalhes que identificam o grupo de snapshots, como o `volumeGroupSnapshotHandle` e os volumeSnapshotHandles individuais existentes no sistema de storage.

Quando uma solicitação de snapshot é recebida, Trident cria o snapshot do grupo de volume no backend. Após a criação do snapshot do grupo de volume, Trident configura um `VolumeGroupSnapshotContent` objeto e, assim, expõe o snapshot à API do Kubernetes.

Kubernetes `CustomResourceDefinition` objetos

Os recursos personalizados do Kubernetes são endpoints na API do Kubernetes definidos pelo administrador e usados para agrupar objetos semelhantes. O Kubernetes oferece suporte à criação de recursos personalizados para armazenar uma coleção de objetos. Você pode obter essas definições de recursos executando `kubectl get crds`.

As definições de recursos personalizados (CRDs) e seus metadados de objeto são armazenados pelo Kubernetes em seu repositório de metadados. Isso elimina a necessidade de um repositório separado para Trident.

Trident usa `CustomResourceDefinition` objetos para preservar a identidade de objetos Trident, como backends Trident, classes de storage Trident e volumes Trident. Esses objetos são gerenciados pelo Trident. Além disso, a estrutura de snapshot de volume CSI introduz alguns CRDs que são necessários para definir snapshots de volume.

Os CRDs são uma construção do Kubernetes. Objetos dos recursos definidos acima são criados pelo Trident. Como um exemplo simples, quando um backend é criado usando `tridentctl`, um objeto CRD correspondente `tridentbackends` é criado para consumo pelo Kubernetes.

Aqui estão alguns pontos a serem lembrados sobre os CRDs do Trident:

- Quando Trident é instalado, um conjunto de CRDs é criado e pode ser usado como qualquer outro tipo de recurso.
- Ao desinstalar Trident usando o comando `tridentctl uninstall`, os pods do Trident são excluídos, mas os CRDs criados não são removidos. Consulte "Desinstalar Trident" para entender como Trident pode ser completamente removido e reconfigurado do zero.

Trident `StorageClass` objetos

Trident cria classes de armazenamento correspondentes para objetos do Kubernetes `StorageClass` que especificam `csi.trident.netapp.io` em seu campo provisioner. O nome da classe de armazenamento corresponde ao do objeto do Kubernetes `StorageClass` que ela representa.

> Com o Kubernetes, esses objetos são criados automaticamente quando um Kubernetes `StorageClass` que usa Trident como provisionador é registrado.

As classes de armazenamento compreendem um conjunto de requisitos para volumes. Trident corresponde esses requisitos com os atributos presentes em cada pool de storage; se eles corresponderem, esse pool de storage é um destino válido para o provisionamento de volumes usando essa classe de armazenamento.

Você pode criar configurações de classes de armazenamento para definir diretamente as classes de armazenamento usando a API REST. No entanto, para implantações do Kubernetes, esperamos que elas sejam criadas ao registrar novos objetos `StorageClass` do Kubernetes.

Objetos de backend Trident

Os backends representam os provedores de armazenamento sobre os quais Trident provisiona volumes; uma única instância do Trident pode gerenciar qualquer número de backends.

Este é um dos dois tipos de objeto que você cria e gerencia sozinho. O outro é o objeto `StorageClass`Kubernetes.

Para obter mais informações sobre como construir esses objetos, consulte "configurando backends".

Trident `StoragePool` objetos

Os pools de storage representam os locais distintos disponíveis para provisionamento em cada backend. Para ONTAP, eles correspondem a agregados em SVMs. Para NetApp HCI/SolidFire, eles correspondem a bandas de QoS especificadas pelo administrador. Cada pool de storage possui um conjunto de atributos de storage distintos, que definem suas características de desempenho e de proteção de dados.

Diferentemente dos outros objetos aqui, os candidatos a pool de storage são sempre descobertos e gerenciados automaticamente.

Trident `Volume` objetos

Volumes são a unidade básica de provisionamento, compreendendo endpoints de backend, como compartilhamentos NFS, e LUNs iSCSI e FC. No Kubernetes, estes correspondem diretamente a `PersistentVolumes`. Ao criar um volume, certifique-se de que ele tenha uma classe de armazenamento, que determina onde esse volume pode ser provisionado, juntamente com um tamanho.

- No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que Trident provisionou.
- Ao excluir um PV com snapshots associados, o volume Trident correspondente é atualizado para o estado **Deleting**. Para que o volume Trident seja excluído, você deve remover os snapshots do volume.

Uma configuração de volume define as propriedades que um volume provisionado deve ter.

Atributo	Tipo	Obrigatório	Descrição
versão	string	não	Versão da API Trident ("1")
nome	string	sim	Nome do volume a ser criado

Atributo	Tipo	Obrigatório	Descrição
storageClass	string	sim	Classe de storage a ser usada ao provisionar o volume
tamanho	string	sim	Tamanho do volume a ser provisionado em bytes
protocolo	string	não	Tipo de protocolo a ser usado; "arquivo" ou "bloco"
internalName	string	não	Nome do objeto no sistema de storage; gerado pelo Trident
cloneSourceVolume	string	não	ontap (nas, san) e solidfire-*: Nome do volume do qual clonar
splitOnClone	string	não	ontap (nas, san): separar o clone de seu progenitor
snapshotPolicy	string	não	ontap-*: política do Snapshot a ser usada
snapshotReserve	string	não	ontap-*: Percentual do volume reservado para snapshots
exportPolicy	string	não	ontap-nas*: política de exportação a ser usada
snapshotDirectory	bool	não	ontap-nas*: se o diretório de snapshot está visível
unixPermissions	string	não	ontap-nas*: Permissões iniciais do UNIX
blockSize	string	não	solidfire-*: Tamanho do bloco/setor
fileSystem	string	não	Tipo de sistema de arquivos
skipRecoveryQueue	string	não	Durante a exclusão de volume, ignore a fila de recuperação no armazenamento e exclua o volume imediatamente.

Trident gera `internalName` ao criar o volume. Isso consiste em duas etapas. Primeiro, ele adiciona o prefixo de storage (o padrão `trident` ou o prefixo na configuração do backend) ao nome do volume, resultando em um nome no formato `<prefix>-<volume-name>`. Em seguida, ele sanitiza o nome, substituindo caracteres não permitidos no backend. Para backends ONTAP, ele substitui hífens por sublinhados (assim, o nome interno se torna `<prefix>_<volume-name>`). Para backends Element, ele substitui sublinhados por hífens.

Você pode usar configurações de volume para provisionar volumes diretamente usando a API REST, mas em

implantações do Kubernetes esperamos que a maioria dos usuários utilize o método padrão do Kubernetes `PersistentVolumeClaim`. Trident cria esse objeto de volume automaticamente como parte do processo de provisionamento.

Trident `Snapshot` objetos

Snapshots são uma cópia pontual de volumes, que podem ser usadas para provisionar novos volumes ou restaurar o estado. No Kubernetes, estes correspondem diretamente a `VolumeSnapshotContent` objetos. Cada snapshot está associado a um volume, que é a fonte dos dados para o snapshot.

Cada objeto `Snapshot` inclui as propriedades listadas abaixo:

Atributo	Tipo	Obrigatório	Descrição
versão	String	Sim	Versão da API Trident ("1")
nome	String	Sim	Nome do objeto de Snapshot do Trident
internalName	String	Sim	Nome do objeto de snapshot do Trident no sistema de storage
volumeName	String	Sim	Nome do Volume Persistente para o qual o snapshot é criado
volumeInternalName	String	Sim	Nome do objeto de volume Trident associado no sistema de storage

No Kubernetes, esses objetos são gerenciados automaticamente. Você pode visualizá-los para ver o que Trident provisionou.

Quando uma solicitação de objeto Kubernetes `VolumeSnapshot` é criada, Trident funciona criando um objeto de snapshot no sistema de storage. O `internalName` desse objeto de snapshot é gerado combinando o prefixo `snapshot-` com o `UID` do `VolumeSnapshot` objeto (por exemplo, `snapshot-e8d8a0ca-9826-11e9-9807-525400f3f660`). `volumeName` e `volumeInternalName` são preenchidos obtendo os detalhes do volume subjacente.

Trident `ResourceQuota` objeto

O Trident daemonset consome uma `system-node-critical` Priority Class—a Priority Class mais alta disponível no Kubernetes—para garantir que Trident possa identificar e limpar volumes durante o desligamento correto do nó e permitir que os pods do Trident daemonset preemptem cargas de trabalho com prioridade mais baixa em clusters com alta pressão de recursos.

Para isso, Trident utiliza um `ResourceQuota` objeto para garantir que a Classe de Prioridade "system-node-critical" no daemonset do Trident seja atendida. Antes da implantação e da criação do daemonset, Trident procura o `ResourceQuota` objeto e, caso não o encontre, o aplica.

Se você precisar de mais controle sobre a Cota de Recursos e a Classe de Prioridade padrão, você pode gerar um `custom.yaml` ou configurar o `ResourceQuota` objeto usando o Helm chart.

O seguinte é um exemplo de um objeto ResourceQuota priorizando o daemonset Trident.

```
apiVersion: <version>
kind: ResourceQuota
metadata:
  name: trident-csi
  labels:
    app: node.csi.trident.netapp.io
spec:
  scopeSelector:
    matchExpressions:
      - operator: In
        scopeName: PriorityClass
        values:
          - system-node-critical
```

Para obter mais informações sobre Resource Quotas, consulte "Kubernetes: cotas de recursos".

Limpe `ResourceQuota` se a instalação falhar

No raro caso em que a instalação falha após o objeto `ResourceQuota` ser criado, tente primeiro "desinstalando" e depois reinstale.

Se isso não funcionar, remova manualmente o `ResourceQuota` objeto.

Remover `ResourceQuota`

Se preferir controlar sua própria alocação de recursos, você pode remover o objeto Trident `ResourceQuota` usando o comando:

```
kubectl delete quota trident-csi -n trident
```

Padrões de Segurança de Pod (PSS) e Restrições de Contexto de Segurança (SCC)

Os padrões de segurança de pods (PSS) e as políticas de segurança de pods (PSP) do Kubernetes definem os níveis de permissão e restringem o comportamento dos pods. OpenShift Security Context Constraints (SCC) definem, de forma semelhante, restrições de pods específicas para o OpenShift Kubernetes Engine. Para fornecer essa personalização, Trident habilita determinadas permissões durante a instalação. As seções a seguir detalham as permissões definidas pelo Trident.

> O PSS substitui as Políticas de Segurança de Pod (PSP). PSP foi descontinuado no Kubernetes v1.21 e será removido na v1.25. Para mais informações, consulte "Kubernetes: segurança".

Contexto de segurança obrigatório do Kubernetes e campos relacionados

Permissão	Descrição
Privilegiado	O CSI exige que os pontos de montagem sejam bidirecionais, o que significa que o pod do nó Trident deve executar um contêiner privilegiado. Para obter mais informações, consulte "Kubernetes: propagação de montagens".
Rede do host	Necessário para o iSCSI daemon. `iscsiadm`Gerencia as montagens iSCSI e usa a rede do host para se comunicar com o iSCSI daemon.
IPC do host	NFS usa comunicação entre processos (IPC) para se comunicar com o NFSD.
PID do host	Necessário para iniciar `rpc-statd` para NFS. Trident consulta os processos do host para determinar se `rpc-statd` está em execução antes de montar volumes NFS.
Capacidades	A `SYS_ADMIN` funcionalidade é fornecida como parte das funcionalidades padrão para contêineres privilegiados. Por exemplo, Docker define essas funcionalidades para contêineres privilegiados: `CapPrm: 0000003fffffffff` `CapEff: 0000003fffffffff`
Seccomp	O perfil Seccomp é sempre "Unconfined" em contêineres privilegiados; portanto, não pode ser ativado no Trident.
SELinux	Em OpenShift, os contêineres privilegiados são executados no `spc_t` ("Super Privileged Container") domínio, e os contêineres não privilegiados são executados no `container_t` domínio. Em `containerd`, com `container-selinux` instalado, todos os contêineres são executados no `spc_t` domínio, o que efetivamente desativa o SELinux. Portanto, Trident não adiciona `seLinuxOptions` aos contêineres.
DAC	Contêineres com privilégios elevados devem ser executados como root. Contêineres sem privilégios elevados são executados como root para acessar os sockets unix necessários pelo CSI.

Padrões de segurança de pod (PSS)

Rótulo	Descrição	Padrão
`pod-security.kubernetes.io/enforce pod-security.kubernetes.io/enforce-version`	Permite que o Trident Controller e os nós sejam admitidos no namespace de instalação. Não altere o rótulo do namespace.	`enforce: privileged enforce-version: <version of the current cluster or highest version of PSS tested.>`

> Alterar os rótulos de namespace pode resultar em pods não sendo agendados, em um "Error creating: …" ou "Warning: trident-csi-…". Se isso acontecer, verifique se o rótulo do namespace para `privileged` foi alterado. Se sim, reinstale Trident.

Políticas de segurança de pods (PSP)

Campo	Descrição	Padrão
`allowPrivilegeEscalation`	Contêineres privilegiados devem permitir a elevação de privilégios.	`true`
`allowedCSIDrivers`	Trident não utiliza volumes efêmeros CSI inline.	Vazio
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contêineres privilegiados recebem todas as capacidades possíveis.	Vazio
`allowedFlexVolumes`	Trident não utiliza um "Driver FlexVolume", portanto eles não estão incluídos na lista de volumes permitidos.	Vazio
`allowedHostPaths`	O pod do nó Trident monta o sistema de arquivos raiz do nó, portanto, não há benefício em configurar esta lista.	Vazio
`allowedProcMountTypes`	Trident não usa nenhum `ProcMountTypes`.	Vazio
`allowedUnsafeSysctls`	Trident não requer nenhum `sysctls` inseguro.	Vazio
`defaultAddCapabilities`	Não é necessário adicionar nenhuma funcionalidade aos contêineres privilegiados.	Vazio
`defaultAllowPrivilegeEscalation`	A permissão para escalonamento de privilégios é gerenciada em cada pod do Trident.	`false`
`forbiddenSysctls`	Não `sysctls` são permitidos.	Vazio
`fsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`

Campo	Descrição	Padrão
`hostIPC`	Montar volumes NFS requer IPC do host para se comunicar com `nfsd`	`true`
`hostNetwork`	O iscsiadm requer que a rede do host se comunique com o daemon iSCSI.	`true`
`hostPID`	É necessário o PID do host para verificar se `rpc-statd` está em execução no nó.	`true`
`hostPorts`	Trident não utiliza nenhuma porta do host.	Vazio
`privileged`	Os pods do nó Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`readOnlyRootFilesystem`	Os pods do nó Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods do nó Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`
`runAsUser`	Os contêineres Trident são executados como root.	`runAsAny`
`runtimeClass`	Trident não usa `RuntimeClasses`.	Vazio
`seLinux`	Trident não configura `seLinuxOptions` porque atualmente existem diferenças na forma como os ambientes de execução de contêineres e as distribuições do Kubernetes lidam com SELinux.	Vazio
`supplementalGroups`	Os contêineres Trident são executados como root.	`RunAsAny`
`volumes`	Os pods Trident requerem esses plugins de volume.	`hostPath, projected, emptyDir`

Restrições de Contexto de Segurança (SCC)

Etiquetas	Descrição	Padrão
`allowHostDirVolumePlugin`	Os pods do nó Trident montam o sistema de arquivos raiz do nó.	`true`

Etiquetas	Descrição	Padrão
`allowHostIPC`	A montagem de volumes NFS requer IPC do host para se comunicar com `nfsd`.	`true`
`allowHostNetwork`	O iscsiadm requer que a rede do host se comunique com o daemon iSCSI.	`true`
`allowHostPID`	É necessário o PID do host para verificar se `rpc-statd` está em execução no nó.	`true`
`allowHostPorts`	Trident não utiliza nenhuma porta do host.	`false`
`allowPrivilegeEscalation`	Contêineres privilegiados devem permitir a elevação de privilégios.	`true`
`allowPrivilegedContainer`	Os pods do nó Trident devem executar um contêiner privilegiado para montar volumes.	`true`
`allowedUnsafeSysctls`	Trident não requer nenhum `sysctls` inseguro.	`none`
`allowedCapabilities`	Os contêineres Trident não privilegiados não exigem mais recursos do que o conjunto padrão e os contêineres privilegiados recebem todas as capacidades possíveis.	Vazio
`defaultAddCapabilities`	Não é necessário adicionar nenhuma funcionalidade aos contêineres privilegiados.	Vazio
`fsGroup`	Os contêineres Trident são executados como root.	`RunAsAny`
`groups`	Este SCC é específico para Trident e está vinculado ao seu usuário.	Vazio
`readOnlyRootFilesystem`	Os pods do nó Trident devem gravar no sistema de arquivos do nó.	`false`
`requiredDropCapabilities`	Os pods do nó Trident executam um contêiner privilegiado e não podem descartar recursos.	`none`
`runAsUser`	Os contêineres Trident são executados como root.	`RunAsAny`

Etiquetas	Descrição	Padrão
`seLinuxContext`	Trident não configura `seLinuxOptions` porque atualmente existem diferenças na forma como os ambientes de execução de contêineres e as distribuições do Kubernetes lidam com SELinux.	Vazio
`seccompProfiles`	Contêineres privilegiados sempre são executados "Unconfined".	Vazio
`supplementalGroups`	Os contêineres Trident são executados como root.	`RunAsAny`
`users`	É fornecida uma entrada para vincular este SCC ao usuário Trident no namespace Trident.	n/a
`volumes`	Os pods Trident requerem esses plugins de volume.	`hostPath, downwardAPI, projected, emptyDir`

Avisos legais

Avisos legais fornecem acesso a declarações de direitos autorais, marcas registradas, patentes e mais.

Direitos autorais

"https://www.netapp.com/company/legal/copyright/"

Marcas registradas

NETAPP, o logotipo da NETAPP e as marcas listadas na página de Marcas Registradas da NetApp são marcas registradas da NetApp, Inc. Outros nomes de empresas e produtos podem ser marcas registradas de seus respectivos proprietários.

"https://www.netapp.com/company/legal/trademarks/"

Patentes

A lista atual de patentes detidas pela NetApp pode ser encontrada em:

https://www.netapp.com/pdf.html?item=/media/11887-patentspage.pdf

Política de privacidade

"https://www.netapp.com/company/legal/privacy-policy/"

Código aberto

Você pode consultar os direitos autorais e as licenças de terceiro usados no software NetApp para Trident no arquivo de avisos de cada versão em https://github.com/NetApp/trident/.

Informações sobre direitos autorais

Copyright © 2026 NetApp, Inc. Todos os direitos reservados. Impresso nos EUA. Nenhuma parte deste documento protegida por direitos autorais pode ser reproduzida de qualquer forma ou por qualquer meio — gráfico, eletrônico ou mecânico, incluindo fotocópia, gravação, gravação em fita ou storage em um sistema de recuperação eletrônica — sem permissão prévia, por escrito, do proprietário dos direitos autorais.

O software derivado do material da NetApp protegido por direitos autorais está sujeito à seguinte licença e isenção de responsabilidade:

ESTE SOFTWARE É FORNECIDO PELA NETAPP "NO PRESENTE ESTADO" E SEM QUAISQUER GARANTIAS EXPRESSAS OU IMPLÍCITAS, INCLUINDO, SEM LIMITAÇÕES, GARANTIAS IMPLÍCITAS DE COMERCIALIZAÇÃO E ADEQUAÇÃO A UM DETERMINADO PROPÓSITO, CONFORME A ISENÇÃO DE RESPONSABILIDADE DESTE DOCUMENTO. EM HIPÓTESE ALGUMA A NETAPP SERÁ RESPONSÁVEL POR QUALQUER DANO DIRETO, INDIRETO, INCIDENTAL, ESPECIAL, EXEMPLAR OU CONSEQUENCIAL (INCLUINDO, SEM LIMITAÇÕES, AQUISIÇÃO DE PRODUTOS OU SERVIÇOS SOBRESSALENTES; PERDA DE USO, DADOS OU LUCROS; OU INTERRUPÇÃO DOS NEGÓCIOS), INDEPENDENTEMENTE DA CAUSA E DO PRINCÍPIO DE RESPONSABILIDADE, SEJA EM CONTRATO, POR RESPONSABILIDADE OBJETIVA OU PREJUÍZO (INCLUINDO NEGLIGÊNCIA OU DE OUTRO MODO), RESULTANTE DO USO DESTE SOFTWARE, MESMO SE ADVERTIDA DA RESPONSABILIDADE DE TAL DANO.

A NetApp reserva-se o direito de alterar quaisquer produtos descritos neste documento, a qualquer momento e sem aviso. A NetApp não assume nenhuma responsabilidade nem obrigação decorrentes do uso dos produtos descritos neste documento, exceto conforme expressamente acordado por escrito pela NetApp. O uso ou a compra deste produto não representam uma licença sob quaisquer direitos de patente, direitos de marca comercial ou quaisquer outros direitos de propriedade intelectual da NetApp.

O produto descrito neste manual pode estar protegido por uma ou mais patentes dos EUA, patentes estrangeiras ou pedidos pendentes.

LEGENDA DE DIREITOS LIMITADOS: o uso, a duplicação ou a divulgação pelo governo estão sujeitos a restrições conforme estabelecido no subparágrafo (b)(3) dos Direitos em Dados Técnicos - Itens Não Comerciais no DFARS 252.227-7013 (fevereiro de 2014) e no FAR 52.227- 19 (dezembro de 2007).

Os dados aqui contidos pertencem a um produto comercial e/ou serviço comercial (conforme definido no FAR 2.101) e são de propriedade da NetApp, Inc. Todos os dados técnicos e software de computador da NetApp fornecidos sob este Contrato são de natureza comercial e desenvolvidos exclusivamente com despesas privadas. O Governo dos EUA tem uma licença mundial limitada, irrevogável, não exclusiva, intransferível e não sublicenciável para usar os Dados que estão relacionados apenas com o suporte e para cumprir os contratos governamentais desse país que determinam o fornecimento de tais Dados. Salvo disposição em contrário no presente documento, não é permitido usar, divulgar, reproduzir, modificar, executar ou exibir os dados sem a aprovação prévia por escrito da NetApp, Inc. Os direitos de licença pertencentes ao governo dos Estados Unidos para o Departamento de Defesa estão limitados aos direitos identificados na cláusula 252.227-7015(b) (fevereiro de 2014) do DFARS.

Informações sobre marcas comerciais

NETAPP, o logotipo NETAPP e as marcas listadas em http://www.netapp.com/TM são marcas comerciais da NetApp, Inc. Outros nomes de produtos e empresas podem ser marcas comerciais de seus respectivos proprietários.